97

TURING



业界专家集体智慧的结晶

97 Things Every Project Manager Should Know

# 项目经理应该知道的97件事

[美] Barbee Davis 编

张科 焦亚超 译

O'REILLY®

人民邮电出版社
POSTS & TELECOM PRESS

# 项目经理应该知道的97件事

如果你管理的项目进展不利，那么看看本书吧！它饱含着业界专家多年实践获得的宝贵知识。本书极具启发性，由97篇短小实用的文章构成，其作者均是世界顶尖的资深项目经理和软件开发人员。关于项目经理应该如何处理各种事务，如管理团队、与项目利益相关者相处、避免会议失控等，他们分享了自己鲜明的观点。

尽管本书强调的是软件项目，但其中涉及的项目管理方法也适用于任何行业所有类型的项目。你既可以逐篇阅读本书的所有内容，也可以只翻阅与你密切相关的那些主题。无论你从事的是否是IT项目，书中的真知灼见都能给你以深刻的启发。

专家们深入思考了如下问题：

- 聪明代码很难维护（戴维·伍德，Zepheira公司的合伙人）
- 每个项目经理都是合同管理者（法比奥·特谢拉·德梅洛，Construtora Norberto Odebrecht公司计划管理经理）
- 报告中挣值与速度两种度量能共存吗（芭比·戴维斯，Davis咨询中心主任）
- 怎样定义"完成"（布赖恩·萨姆-博登，作家，软件架构师）
- 做实际工作的人才是最好的估算人员（乔·泽尼维奇，ThoughtWorks公司高级项目经理）
- 如何发现优秀的IT开发人员（詹姆斯·格雷厄姆，独立管理咨询顾问）
- 一件交付任务需由一人负总责（艾伦·格林布拉特，Sciova公司的首席执行官）



**Barbee Davis** 文学硕士，人力资源专家，项目管理专业人员，美国管理协会*Community Post*半月专栏撰稿人。她自己的戴维斯咨询公司专门从事人力资源培训和咨询服务。

封面设计：Mark Paglietti　张健

图灵网站：www.turingbook.com 热线：(010)51095186转604
反馈/投稿/推荐信箱：contact@turingbook.com
有奖勘误：debug@turingbook.com

分类建议 计算机/项目管理

O'REILLY®
oreilly.com.cn




人民邮电出版社网址：www.ptpress.com.cn

O'Reilly Media, Inc.授权人民邮电出版社出版

此简体中文版仅限于中国大陆（不包含中国香港、澳门特别行政区和中国台湾地区）销售发行
This Authorized Edition for sale only in the territory of People's Republic of China (excluding Hong Kong, Macao and Taiwan)

ISBN 978-7-115-25100-8




ISBN 978-7-115-25100-8
定价：39.00元

TURING

# 项目经理应该知道的97件事

## 97 Things Every Project Manager Should Know


［美］Barbee Davis 编

张科　焦亚超 译





人民邮电出版社

北　京

**图书在版编目（CIP）数据**

项目经理应该知道的97件事 /（美）戴维斯(Davis,B.）编；张科，焦亚超译. -- 北京：人民邮电出版社，2011.8

书名原文：97 Things Every Project Manager Should Know

ISBN 978-7-115-25100-8

Ⅰ. ①项… Ⅱ. ①戴… ②张… ③焦… Ⅲ. ①软件开发－项目管理 Ⅳ. ①TP311.52

中国版本图书馆CIP数据核字(2011)第053112号

## 内 容 提 要

本书是集体智慧的结晶，是来自世界各地的具有成功项目管理经验的项目经理、软件开发人员和其他职业领域的专家集体创作的。书中，这个领域里活跃分子分享了他们多年来积累的经验和秘诀，作者将诸多成功项目经理的经验提炼为97个方法，读者可以随意捡起其中一个急需的方法。

本书是一本项目经理的实战宝典，项目开发人员、软件经理等项目相关人员也能从中获得有益的指导。

**项目经理应该知道的97件事**

◆ 编　[美] Barbee Davis
译　张 科 焦亚超
责任编辑　明永玲

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街14号
邮编 100061　电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京隆昌伟业印刷有限公司印刷

◆ 开本：700×1000 1/16
印张：15
字数：227千字　2011年8月第1版
印数：1－4 000册　2011年8月北京第1次印刷

著作权合同登记号　图字：01-2010-4016号

ISBN 978-7-115-25100-8

定价：39.00元

读者服务热线：(010)51095186转604　印装质量热线：(010)67129223

反盗版热线：(010)67171154

# O'Reilly Media, Inc. 介绍

O'Reilly Media 通过图书、在线服务、杂志、调查研究和会议等方式传播创新者的知识。自 1978 年开始 O'Reilly 一直都是发展前沿的见证者和推动者。超级极客正在开创未来，我们关注着真正重要的技术趋势，通过放大那些"微弱的信号"来刺激社会对新科技的采用。作为技术社区中活跃的参与者，O'Reilly 的发展充满着对创新的倡导、创造和发扬光大。

作为出版商 O'Reilly 为软件开发人员带来革命性的"动物书"，创造了第一个商业网站（GNN），组织开放源代码峰会以至于开源软件运动以此命名，通过创立 Make 杂志成为 DIY 革命的主要先锋，公司一如既往地用各种方式和渠道连接人们和他们所需要的信息。O'Reilly 的会议和峰会聚集了超级极客和高瞻远瞩的商业领袖，共同描绘将开创新产业的革命性思想。作为技术人士获取信息的选择 O'Reilly 现在还将先锋专家的知识传递给普通计算机用户。无论是通过印刷书籍、在线服务或者面授课程，每一项 O'Reilly 的产品都反映了公司不可动摇的信念——信息是激发创新的力量。

## 业界评论

"O'Reilly Radar 博客有口皆碑。"

——Wired

"O'Reilly 凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。"

——Business 2.0

"O'Reilly Conference 是聚集关键思想领袖的绝对典范。"

——CRN

"一本 O'Reilly 的书就代表一个有用、有前途、需要学习的主题。"

——Irish Times

“Tim 是位少有的商人，他不光放眼于最长远、最广阔的视野并且切实地按照 Yogi Berra 的建议去做了：“如果你在路上遇到岔路口，走小路（岔路）。”回顾过去 Tim 似乎每一次都选择了小路，而且有几次都是一闪即瞬的机会，尽管大路也不错。”

——Linux Journal



# 译 者 序

软件行业的变化日新月异，促使从业人员不断学习更加先进的知识和技术，行业激烈的竞争也促使软件项目经理必须全方位考虑以保证项目的成功。

本书汇集了世界各地软件行业的专业人士论述软件项目管理的97篇短文，阐明了软件项目经理应该注意的各种问题，如谈敏捷方法方面的《引进一种更敏捷的沟通系统》，谈国际软件开发的《一词不慎坏大事》，谈管理人员和团队的《为团队增添人才而非技能》，谈需求管理的《让项目发起人自己写需求》，谈项目管理的《项目管理即问题管理》，谈沟通的《会议写不出代码》，谈利益相关者管理的《随时间滚动》，谈项目流程的《三位一体的项目管理》，谈项目需求的《要简单，不要复杂》，还有涉及最终用户的《尽早让用户参与》，涉及采购问题的《买现成的软件》，涉及自我管理的《你并不是非比寻常的》，涉及Web开发的《就是现在而非马上》，等等。每一个方面都有不同专业人士的看法，这些都是他们经过实践检验的经验，对从事软件行业的人员特别是软件项目经理来说是很有指导和启发意义的。

牛顿说过："如果说我看得比别人更远些，那是因为我站在巨人的肩膀上。"这说明了前人经验的重要性。本书为软件项目经理提供了一个很好的参考。当然遇到具体问题还是要具体分析，实事求是，不能一味照搬照抄。

在翻译本书的过程中，我越来越深刻地体会到软件项目经理所扮演的角色其实就是一个协调者，打个比方，就像是"润滑剂"。当项目利益相关者、客户、项目团队开发人员等发生利害冲突时，软件项目经理要协调好各方面的利益和关系，使整个系统润滑，减少摩擦和阻力，从而使项目和整个团队向前发展，使各方面利益最大化。当然还会有企业文化、敏捷软件开发方法、自我管理等多个方面需要了解。软件项目经理只有运用好这些方法来指导实践，协调好各方利益，才能促成项目的最后成功。

# 前　　言

在理论上，创造一个新产品或者提出一种新方法很简单。但在实践中，我们这些为了生存而去真正操作的人知道，它正变得越来越混乱。

本书是集体智慧的结晶，是来自世界各地的具有成功项目管理经验的项目经理、软件开发人员和其他职业领域的专家集体创作的。从开发产品到管理公司的项目，他们与你分享了许许多多你应该知道的重要思想。

传统项目管理书籍偏重理论，而在这本书里，本领域里的活跃分子分享了他们多年来积累的经验和秘诀。利用他们实用的建议，解决问题并圆满完成项目，不仅能够改进产品，而且还能丰富你的个人经验。

在和一些主动实践者交谈时，我发现在各种项目中，不管是软件开发人员、研发化学师、建筑队长，还是其他行业的技术专家，都在越来越积极地发出自己的声音。当然，这个参与人数越来越多的行业肯定也少不了用户和利益相关者的身影。尽管项目中大量合作十分可贵，但这种合作无疑也成倍地增加了完成工作的复杂性。

有趣的是，在编著这本书时，我发现无论什么行业、什么项目角色或哪个国家和地区，所有在项目中承担责任的人都面临着同样的挑战。而令人高兴的是，这些来自世界各地的作者都愿意分享他们的想法，帮助更多的人应对这些挑战。他们不仅代表项目经理，而且也代表项目中那些新技术利益相关者的声音，这可是一个好机会，提前听听这些人都有什么想法，他们最关心什么，你就可以未雨绸缪了。

我坚信“共享的知识就是力量”。本书是由来自美国 29 个地方和 12 个其他国家的作者合作撰写而成的。这些作者无私地奉献出了他们的想法和建议，通过传授这些技巧，力求帮助这个领域中的其他人进步和成功。尽管大家都很忙，但他们仍然愿意花时间分享自己的宝贵经验，这足以说明大家对协作的价值都

充满信心。

## 使用许可

本书中的每一篇文章都具有开源的性质，它们的英文原版都已经基于创意共享许可（Creative Commons, Attribution 3）在网上公开[①]。换句话说，你可以在自己的工作中使用每一篇文章，但必须指出文章的出处和作者。曾经也有人尝试过写作开源图书但除了几本以外，大都以失败而告终。我想那主要是因为在没有模块化的情况下，一个人不容易对整个项目有太大贡献。而本书则成功地做到了这一点：每一位作者的建议都独立成篇，既与全书浑然一体，单独阅读也完整可用。

## 我们的联系方式

请把对本书的评论和问题发给出版社。

美国：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）

奥莱利技术咨询（北京）有限公司

O'Reilly 的每一本书都有专属网页，你可以在那儿找到关于本书的相关信息，包括勘误表、示例代码以及其他的信息。本书的网站地址是：

http://www.oreilly.com/catalog/9780596804169/

中文书

http://www.oreilly.com.cn/book.php?bn=9787115251008

① 参见 http://pm.97things.oreilly.com/wiki/。——编者注

对于本书的评论和技术性的问题，请发送电子邮件到：

bookquestions@oreilly.com

关于本书的更多信息、会议、资源中心和网络，请访问以下网站：

http://www.oreilly.com
http://www.oreilly.com.cn

## Safari®在线图书

当你在喜爱的技术书封面上看到这个图标时，就意味着你可以通过 O' Reilly Network Safari Bookshelf 在线浏览本书。

Safari 比电子书更好，它是一个虚拟图书馆，可以让你轻松搜索上千种高端技术图书、复制粘贴代码示例、下载章节，并在需要最准确、最新的信息时快速找到答案。请到 http://my.safaribooksonline.com 免费注册。

# 目　　录

技巧分类……XIII
译者序……XIX
前言……XX
致谢……1

1. 尽早让用户参与……2
芭比·戴维斯

2. 避免打地鼠式开发……4
温卡特·苏布拉马尼亚姆

3. 一词不慎坏大事……6
帕维尔·西姆沙

4. 让项目发起人自己写需求……8
竹家美代子

5. 要简单，不要复杂……10
斯科特·戴维斯

6. 偿还你的技术债……12
布莱恩·斯莱滕

7. 为团队增添人才而非技能……14
理查德·谢里登

8. 西蒙，保持简单……16
克利斯那·卡达利

9. 你并不是非比寻常的……18
贾里德·理查森

10. 随时间滚动……20
金·麦科马克

11. 你们的问题，我不买单 ········ 22
兰迪·卢米斯
12. 如何发现优秀的 IT 开发人员 ········ 24
詹姆斯·格雷尼姆
13. 优秀与普通的天壤之别 ········ 26
尼尔·福特
14. 规模决定一切 ········ 28
阿努潘·昆杜
15. 记录工作流程，然后严格执行 ········ 30
蒙特·戴维斯
16. 剔除多余的流程 ········ 32
纳雷什·贾因
17. 矛盾体的需求说明书 ········ 34
艾伦·格林布拉特
18. 商业价值始终是衡量成功的标准 ········ 36
芭比·戴维斯
19. 不要总因项目放弃休假 ········ 38
乔·泽尼维奇
20. 集中精力 ········ 40
詹姆斯·利
21. 项目管理即问题管理 ········ 42
洛林·昂格尔
22. 授权：蒂姆的故事 ········ 44
肯·赛普
23. 聪明代码很难维护 ········ 46
戴维·伍德
24. 掌控人的因素 ········ 48
詹姆斯·格雷尼姆
25. 使用维基 ········ 50
阿德里安·威布尔

26. 缺失的环节 52
保罗·瓦戈纳

27. 估算，估算，再估算 54
理查德·谢里登

28. 项目管理办公室在前进 56
安杰洛·瓦尔

29. 重苦劳，更重功劳 58
温卡特·苏布拉马尼亚姆

30. 软件的失败是组织的失败 60
布莱恩·斯莱滕

31. 来自另一端的声音 62
马蒂·斯科莫

32. 保持洞察力 64
詹姆斯·格雷厄姆

33. 怎样定义“完成” 66
布赖恩·萨姆—博登

34. 60/60 定律 68
戴维·伍德

35. 遭遇敌人……敌人就是我们自己 70
芭比·戴维斯

36. 工作循环 72
詹姆斯·利

37. 先照顾好自己 74
哈里·塔克

38. 别指望开会写出代码 76
威廉·J.米尔斯

39. 绘制变化进程图 78
凯西·麦克杜格尔

40. 达成共识 80
戴维·迪亚兹·卡斯特罗

41. 依据现实制定计划……82
克雷格·莱特维克

42. 完美实现的谬误……84
戴维·伍德

43. 敏捷的沟通系统……86
布赖恩·萨姆-博登

44. 不要崇拜方法论……88
法比奥·特谢拉·德梅洛

45. 电子表格解决不了人的问题……90
阿努潘·昆杜

46. 一件交付任务需由一人负总责……92
艾伦·格林布拉特

47. 知识完善的谬论……94
戴维·伍德

48. 培养团队跑马拉松，而不是冲刺……96
纳雷什·贾因

49. 三位一体的项目管理……98
保罗·瓦戈纳

50. 路线图：最近我们为你做了什么……100
凯西·麦克杜格尔

51. 项目范围说明的重要性……102
金·海德曼

52. 愿景与预期结果保持一致……104
戴维·迪亚兹·卡斯特罗

53. 艾丽丝不是美国人了……106
芭比·戴维斯

54. 避免合同纠纷……108
乔治·格拉伯特

55. 评估什么，就得到什么……110
纳雷什·贾因

56. 他山之石，可以攻玉 112
保罗·贾马尔沃

57. 要现在不要马上 114
斯科特·戴维斯

58. 速度就是生命，越快越好 116
马特·"布姆"·丹尼尔

59. 激发团队士气 118
戴维·博克

60. 项目要依靠团队合作 120
莱利奥·瓦芮拉

61. 为团队服务 122
卡伦·吉利森

62. 大圆球谬论 124
戴维·伍德

63. 应对危机 126
詹姆斯·格雷尼姆

64. 了解集成要点 128
蒙特·戴维斯

65. 分布式项目要积极促进沟通 130
阿努潘·昆杜

66. 在开始时就要胸有成竹 132
路易斯·托雷斯

67. 清晰的条款，长久的友谊 134
马泰奥·贝基

68. 做实际工作的人才是最好的估算人员 136
乔·泽尼维奇

69. 沟通最关键 138
杰纳迪·米罗诺夫

70. 项目就是对解决方案的追求 140
辛西娅·伯格

71. 傻瓜，人最关键 ······ 142
阿德里安·威布尔

72. 文档是手段而非目的 ······ 144
帕特里克·夸

73. 报告中挣值与速度两种度量能共存吗 ······ 146
芭比·戴维斯

74. 范围改变经常发生，要适应它 ······ 148
帕维尔·西姆沙

75. 买现成的软件 ······ 150
埃尔纳尼·马奎斯·达席尔瓦

76. 三类项目赞助人 ······ 152
乔治·格拉伯特

77. 该少于承诺还是多交付 ······ 154
乔·泽尼维奇

78. 每个项目经理都是合同管理者 ······ 156
法比奥·特谢拉·德梅洛

79. 重要，但不紧急 ······ 158
亚历克斯·米勒

80. 讲授流程 ······ 160
理查德·谢里登

81. 状态的假象 ······ 162
尤迪·达罕

82. 他们到底想听什么 ······ 164
玛莎·勒加雷

83. 团队士气金不换 ······ 166
戴维·博克

84. 让利益相关者全程参与 ······ 168
卢克曼·拉瓦尔

85. 计划的价值 ······ 170
德里·齐美尔

86. 不要总是扮演“信使”…… 172
马特·萨克斯科

87. 有效管理交付产品…… 174
埃尔纳尼·马奎斯·达席尔瓦

88. 我们只是项目经理，不是超级英雄…… 176
安吉妮·肖克–史密斯

89. 增加交流：时常召开即时会议…… 178
理查德·谢里登

90. 用灵活性简化项目管理…… 180
克利斯那·卡达利

91. Web 为现在指明了道路…… 182
戴维·伍德

92. 开发者厌烦状态报告，经理们却喜欢…… 184
帕维尔·西姆沙

93. 你没有控制住…… 186
帕特里克·夸

94. 分享观点…… 188
贾里德·理查森

95. 善于支持的组织就能获得成功…… 190
辛西娅·伯格

96. 建立项目管理控制…… 192
埃尔纳尼·马奎斯·达席尔瓦

97. 我讨厌你的网站的 9.7 个原因…… 194
芭比·戴维斯

撰稿人…… 196

# 技巧分类

## 敏捷方法


尽早让用户参与……2
西蒙，保持简单……16
随时间滚动……20
如何发现优秀的IT开发人员……24
不要总因项目放弃休假……38
授权：蒂姆的故事……44
怎样定义“完成”……66
工作循环……72
敏捷的沟通系统……86
知识完善的谬论……94
要现在不要马上……114
为团队服务……122
做实际工作的人才是最好的估算人员……136
报告中挣值与速度两种度量能共存吗……146
增加交流：时常召开即时会议……178


## 软件开发


尽早让用户参与……2
避免打地鼠式开发……4
一词不慎坏大事……6
要简单，不要复杂……10
偿还你的技术债……12
剔除多余的流程……32
集中精力……40
聪明代码很难维护……46
项目管理办公室在前进……56
软件的失败是组织的失败……60
来自另一端的声音……62
怎样定义“完成”……66
60/60定律……68

工作循环……72
完美实现的谬误……84
知识完善的谬论……94
愿景与预期成果保持一致……104
艾丽丝不是美国人……106
要现在不要马上……114
大圆球谬论……124
了解集成要点……128
范围改变经常发生，要适应它……148
买现成的软件……150
用灵活性简化项目管理……180
Web为现在指明了道路……182
开发者厌烦状态报告，经理们却喜欢……184

## 管理人员和团队

避免打地鼠式开发……4
为团队增添人才而非技能……14
你并不是非比寻常的……18
如何发现优秀的IT开发人员……24
优秀与普通的天壤之别……26
商业价值始终是衡量成功的标准……36
授权：蒂姆的故事……44
聪明代码很难维护……46
掌控人的因素……48
缺失的环节……52
估算，估算，再估算……54
重苦劳，更重功劳……58
软件的失败是组织的失败……60
遭遇敌人……敌人就是我们自己……70
工作循环……72
别指望开会写出代码……76
绘制变化进程图……78
电子表格解决不了人的问题……90
一件交付任务需由一人负总责……92
培养团队跑马拉松，而不是冲刺……96
三位一体的项目管理……98
愿景与预期成果保持一致……104
评估什么，就得到什么……110

激发团队士气……118
项目要依靠团队合作……120
做实际工作的人才是最好的估算人员……136
傻瓜，人最关键……142
讲授流程……160
状态的假象……162
团队士气金不换……166

## 国际化问题和分布式团队

一词不慎坏大事……6
让项目发起人自己写需求……8
矛盾体的需求说明书……34
达成共识……80
不要崇拜方法论……88
艾丽丝不是美国人了……106
分布式项目要积极促进沟通……130
沟通最关键……138
开发者厌烦状态报告，经理们却喜欢……184
你没有控制住……186
分享观点……188

## 管理项目

规模决定一切……28
记录工作流程，然后严格执行……30
项目管理即问题管理……42
使用维基……50
怎样定义“完成”……66
60/60定律……68
达成共识……80
依据现实制定计划……82
应对危机……126
在开始时就要胸有成竹……132
文档是手段而非目的……144
该少于承诺还是多交付……154
重要，但不紧急……158
有效管理交付产品……174

## 沟通

优秀与普通的天壤之别……26
使用维基……50

项目管理办公室在前进……56
别指望开会写出代码……76
敏捷的沟通系统……86
电子表格解决不了人的问题……90
路线图：最近我们为你做了什么……100
分布式项目要积极促进沟通……130
沟通最关键……138
傻瓜，人最关键……142
三类项目赞助人……152
每个项目经理都是合同管理者……156
他们到底想听什么……164
让利益相关者全程参与……168
不要总是扮演“信使”……172
增加交流：时常召开即时会议……178

## 利益相关者管理

让项目发起人自己写需求……8
西蒙，保持简单……16
随时间滚动……20
你们的问题，我不买单……22
商业价值始终是衡量成功的标准……36
绘制变化进程图……78
路线图：最近我们为你做了什么……100
项目范围说明的重要性……102
避免合同纠纷……108
项目就是对解决方案的追求……140
三类项目赞助人……152
该少于承诺还是多交付……154
讲授流程……160
让利益相关者全程参与……168
善于支持的组织就能获得成功……190
建立项目管理控制……192

## 项目流程

剔除多余的流程……32
依据现实制定计划……82
不要崇拜方法论……88
一件交付任务需由一人负总责……92
三位一体的项目管理……98

项目范围说明的重要性……102
评估什么，就得到什么……110
他山之石，可以攻玉……112
应对危机……126
了解集成要点……128
在开始时就要胸有成竹……132
清晰的条款，长久的友谊……134
项目就是对解决方案的追求……140
文档是手段而非目的……144
报告中挣值与速度两种度量能共存吗……146
范围改变经常发生，要适应它……148
状态的假象……162
计划的价值……170
有效管理交付产品……174
用灵活性简化项目管理……180
建立项目管理控制……192

## 项目需求

让项目发起人自己写需求……8
要简单，不要复杂……10
西蒙，保持简单……16
矛盾体的需求说明书……34
买现成的软件……150

## 最终用户

尽早让用户参与……2
要简单，不要复杂……10
记录工作流程，然后严格执行……30
来自另一端的声音……62
保持洞察力……64
我讨厌你的网站的9.7个原因……194

## 采购问题

你们的问题，我不买单……22
避免合同纠纷……108
买现成的软件……150
每个项目经理都是合同管理者……156

## 自我管理

你并不是非比寻常的……18

不要总因项目放弃休假 38
集中精力 40
项目管理即问题管理 42
重苦劳，更重功劳 58
保持洞察力 64
遭遇敌人……敌人就是我们自己 70
先照顾好自己 74
培养团队跑马拉松，而不是冲刺 96
他山之石，可以攻玉 112
速度就是生命，越快越好 116
激发团队士气 118
为团队服务 122
重要，但不紧急 158
团队士气金不换 166
不要总是扮演“信使” 172
我们只是项目经理，不是超级英雄 176
你没有控制住 186
分享观点 188
善于支持的组织就能获得成功 190

## Web 开发

随时间滚动 20
规模决定一切 28
来自另一端的声音 62
要现在不要马上 114
Web为现在指明了道路 182
我讨厌你的网站的9.7个原因 194



# 致谢

编写本书的想法不是凭空而来的。为此，我要感谢为本书的理念和最终完成作出贡献的人。

我想要感谢丛书编辑理查德·蒙森-黑费尔，我是在帮助杰伊·齐默尔曼组织 No Fluff Just Stuff 大会时遇到他的。当他发现我关注项目管理和软件开发之后，就建议我为他的“97 件事”系列写一本书，书名就叫《项目经理应该知道的 97 件事》，作为他自己的《软件架构师应该知道的 97 件事》的姊妹篇。

奥莱利公司网站上有一个公开的维基，世界各地有兴趣的人都可以参与进来。我深深地感谢那些为本书奉献了他们时间和建议的人。

奥莱利公司开放出版、察纳雅言，支持采用一种或多或少未经检验的方式写书，这一点非常值得尊敬。同时，我也非常赞赏他们同意将图书的内容开放（基于 Creative Commons，Attribution 3）的决定。我要感谢奥莱利公司的 Mike Loukides、Rachel Monaghan、Ed Stephenson 和 Laurel Ackerman。没有他们的帮助和指导，这个项目不可能完成。

奥莱利还会陆续出版其他“97 件事”系列图书，这一系列旨在创造一个独一无二的新品牌，特点是集合一个领域中多位专家的实践经验，并以协作方式完成。目前已经出版的项目管理、软件开发和数据架构图书仅仅是该系列的一小部分。

# 1. 尽早让用户参与



**芭比・戴维斯（Barbee Davis）**
文学硕士，PHR[①]，PMP
美国内布拉斯加州奥马哈市

以往的软件开发模式是先了解用户的要求，然后再在极度神秘的环境下进行编程测试。毕竟，用户根本不知道我们在做什么，对吧？直至项目结束，我们的魔术师才会匆匆登场，揭去魔法布，然后期望用户会对我们卓越的产品惊叹不已。然而用户此时的通常反应是："咳，好吧，我知道你们花了很多功夫，但我们真正需要的是……"

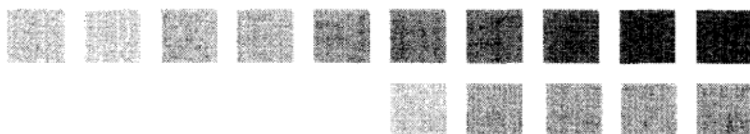
今天，项目成功的秘诀就是尽早向用户展示任何可以展示的东西。如果能在项目启动早期，而不是当整个项目都结束的时候，就能发现项目中存在的问题，那该有多好啊！

随着项目时间的推进，变更项目的成本越来越高。重新编码、重新测试以及改进当前软件的时间，加上整合外围代码进行集成测试所花费的时间都会大大拖延项目进度。如果变化非常重大，还可能危及时间和成本底线，需要经过变更控制委员会一个漫长的批准程序。

在一些细小环节上做出的编程决策，或许对于软件开发人员和项目经理而言道理十足，可当软件投入使用后却有可能给用户造成巨大的混乱。

我知道曾经有一个大型的培训公司，花了500万美元重新设计它的订购软件系统。此前，项目编号同订购产品之间的匹配存在一定逻辑性。例如，4125可能是学生手册，4225则是配套的学生练习盘，4325可以代表教师手册，4425则是营销宣传时使用的课程大纲等。你可以在同一屏幕上订购4X25系列的所有项目。

① PHR，即Professional in Human Resources（人力资源专家），是由美国认证协会（American Certification Institute）推出的在国际范围内比较权威的人力资源管理知识体系认证资格。——编者注



每天，全球 140 个地方的行政助理一遍遍地反复订购同类材料，并很快记住了项目编号。一旦知道了学生手册的编号，他们无须查看就可以立即键入其他项目的编号，这样一来，订购过程十分快捷。

在重新设计时，不知何故，这个项目团队居然忘记考虑实际情况下真人是如何进行订购的。新的设计方式中，项目之间没有逻辑关系。项目 6358 可能就是 4125 曾经代表的学生手册，而其配套的学生练习盘现在是 8872，而同一类教师手册又是 3392。

现在，不仅用户不得不逐项查询并试着“忘记”旧的编号和系统，而且同类产品的不同项目也会分别出现在不同的页面上。

行政助理们很愤怒。订购过程慢得就像是蜗牛在蠕动。该项目最后远远突破了它的时间和成本底线。

作为项目经理，你应该尽早并经常让软件开发人员和用户交谈。



# 2. 避免打地鼠式开发



**温卡特·苏布拉马尼亚姆（Venkat Subramaniam）**
美国科罗拉多州布鲁姆菲尔德市

软件项目经理经常面临及早交付产品的巨大压力。时间是关键。你如何才能快速完成任务呢？

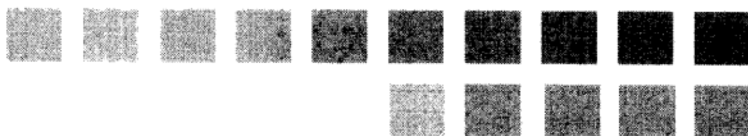
假设你的团队有两名程序员，伯尼和罗布。两人都很能干，知识面相同，编程语言技巧也相当。你发现在开发过程中伯尼实现软件功能的速度远远超过罗布。

当伯尼着力于快速完成编程时，罗布正花时间写代码并对其进行重构[①]。罗布对变量和方法的命名更擅长。一旦写的程序能够运行起来，他就把这个程序分成几小块。现在他要写测试来确保该程序的每一块都能按照他的意图运行。当他觉得结果比较满意时才宣称实现了功能。

但是假设你并不知道上述这些细节。如果你只看他们谁先宣称实现了功能，那么很明显伯尼表现得更好，对吗？

几个星期之后，你向客户演示这些功能。像往常一样，顾客喜欢你已经完成的功能，但现在需要你做一些改动和完善。你让软件开发人员修改这些代码。当你把新改进的功能带回给客户时，他们试用了罗布实现的功能，对他做出的改动十分满意。

① 重构就是改动代码，完善其内部组成结构而不改变其外部功能。它改进软件产品设计。重构代码是回过头去完善以前仓促创建并测试的某项可用的功能。现在需要进一步对该功能进行内部改进，以便长期使用，也使日后增加功能更为方便。



遗憾的是他们发现伯尼实现的功能有些奇怪。当伯尼编写好新的功能后，一些以前能使用的功能现在却不能用了。客户把这些作为缺陷标记出来，然后你让伯尼来修改。客户又一次测试这些功能，结果后来新增的功能也不能用了。这到底是咋回事呢？

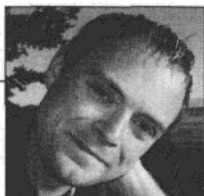
如果你有孩子，就会知道发生了什么。伯尼创建的是一个打地鼠式的应用程序。打地鼠是一种游戏，孩子们拿着一个木棒敲打几个孔中随意弹出的地鼠，他们会很惊奇接下来是哪个地鼠弹出来，而且还乐此不疲。然而，在修改应用程序的时候如果总有坏代码不知从何处随意弹出，那可不是好玩的事情。那会令人沮丧，结果也难以预料，并且它会减慢产品开发速度。伯尼一开始就全速冲刺，只是南辕北辙了。

尽管罗布从一开始就表现得较慢，但实际上他编写的代码更胜一筹。事实证明，他的节奏是可持续发展的。由于最初编写的代码就有较好的质量，所以，他才能更快地做出可行的改动。而且他早期编写的测试能随时带给他反馈信息，让他了解自己新写的代码是否与原有应用程序的其他部分兼容。

当估计某一功能的实现时间时，不要只考虑最初写代码所花费的时间，还要加上提升、调整和改进这些代码所需的时间。写高质量的代码和测试都需要花时间。从短期来看，这似乎是一种损失，然而它带来的却是长期收获。

问问自己，你是想要速度还是想尽情享受可持续发展的乐趣。

# 3. 一词不慎坏大事



**帕维尔·西姆沙（Pavel Simsa）**
PMP
美国华盛顿州贝尔维尤市

哪一个词会让你错过最后期限？答案是“任何一个词”。在开发一个将以非英语语言发布的产品时，你将给项目带来许多新的风险和限制。

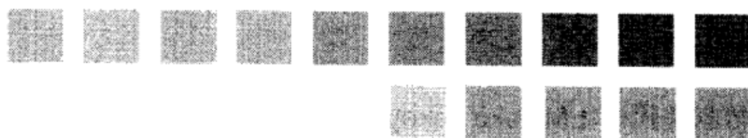
有些风险和限制是技术性的，且显而易见。例如，如果你的产品将投放日本，那么它必须支持适当的字体。如果不支持，那么，即使英文版运行得很完美，日文版的产品也无法运行。但是，字体兼容性不受你的控制。你和团队需要特别注意的倒是一些特殊的翻译译法，并在编码前就要考虑到这些因素，确保开发活动遵循有关国际标准，消除这类问题。

仅仅是转换成其他语言版本也会影响到你何时做何种决定。通常情况下，产品本地化（日语、瑞典语和德语等）与英语版本的开发是同时进行的，只是具有一定的滞后。滞后的时间可以是几天，几周，甚至几个月。尽管如此，在某个时间点，外文版的翻译必须要“赶上”英文版。

在测试和审查阶段，你需要确保：

- 英文版里的内容都可以被准确翻译过来；
- 翻译过来的内容同英文版是真正相符的；
- 翻译版产品的运行没有瑕疵。

这里有一个问题。这三项事情可能是在英文版完成并批准后才进行测试的。在测试和审查本地化的版本时，你总会发现不止一个具有挑战性的问题无法得到解决，除非回头去改变英文版产品。



然而，要知道，在英文产品里最后一分钟所做的一个相对简单和低风险的改变，如改写句子（这只需要几秒钟来编码），却往往需要好几天时间来运行和重新测试所有本地化的版本。

这可能需要额外花费数千美元，尤其是当你正和一家国外的公司签订翻译合约时。经验不足的软件开发项目经理常犯的这个错误很简单。他们就是低估了对英文版进行意外更改所造成的影响有多严重。

为了预防这种情况发生，你可以采取以下两项主要措施。

- 在进度表最后加一个“本地化缓冲区”。进度表的截止日期意味着与项目计划中的英语产品相关的所有工作都必须在这个有效期限内结束。在目标结束日期之后做的任何改动，都必须符合非常具体、非常严格的标准，只有达到这个标准才可以“进入”返工队列。这个版本的任何变化都不可避免地引起其他外文版作出相应变化。
- 把这些任务排序，让功能的质量控制和英文文本的质量审查分开来做。这其实很简单，甚至可以复制所有英文文本到一个电子表格里去进行校对。这样一来，如果有措辞不清楚的情况，我们就能立即发现它，而不必等到测试可运行的产品时才发现它。于是我们可以提前作出必要的改动，可能就不需要对所有其他的语言版本都进行返工了。

# 4. 让项目发起人自己写需求



**竹家美代子（Miyoko Takeya）**
PMP
日本东京

项目失败并不是只有美国公司才遇到的问题。根据几年前一家日本领先 IT 杂志社进行的调查，如果按照质量、成本和交付期的标准来衡量，日本企业实施的项目中，超过 75% 都被认为是失败的。

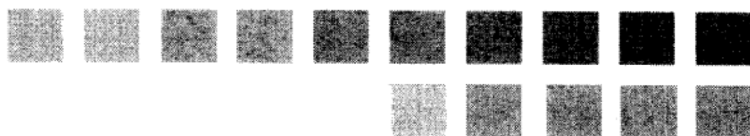
跟大多数其他国家一样，在日本，项目不能达标的首要原因几乎总是需求定义太差。处境最危险的公司是那些业务分析能力较差的公司。由他们来做软件开发这样的技术项目时，成功就被委婉地归为“不可能发生的”事项。这一结果表明，要发现、识别并定义一个软件项目的真实需求有多么困难。

既然是这样困难，很多项目所有人，如客户、项目发起人或公司主管，就期望项目经理能自己定义和细化这些对软件的要求。项目所有人很少会提供指导或明确他们的需求。既然是一个软件项目，而自己可能又不懂软件开发，所以，他们认为没必要来亲自定义自己的期望。

而对于软件项目经理而言，他通常没有权力或时间来自己发现、选择项目需求并排定优先级，尤其是当该项目可能涉及多个利益团体时，大家对软件完成后应实现的功能可能会有相互冲突的设想。

项目经理要在项目实施前花时间与资助该项目的人交流，帮他们准确定义各自想要的功能。要能够迅速完成项目且只有少数错误，而且需要的预算尽可能地少，这难道不是最重要吗？但你又不能一箭三雕。哪些资源和技能对创建所需软件而言是至关重要的？资助人是否给团队提供了这些资源？

软件如何用于运行基础架构或为公司赚钱？时间期限是切实可行的吗？这些



时间限制是不是被写进客户合同，是否绑定到某个重要的节假日，是否据此精心制作了一份营销计划？

如果在需求定义阶段没有认真、具体地考虑这个项目要创建什么，那么，这个项目可是岌岌可危了。请记住，项目所有人需要传达的是他们想要这个软件做什么，而不是程序员将如何着手生成这一结果。

说服项目所有人，他们必须自始至终参与这个过程。步调一致的需求计划可以明确地将商业意义、项目目标和项目结果联系到一起。否则，该项目不能产生他们所期待的令人满意的结果。

失败的软件项目对项目所有人伤害最大，因为是他们资助了这个项目，是他们一直期待着能靠这个软件赚回投资。



# 5. 要简单，不要复杂



**斯科特·戴维斯（Scott Davis）**
美国科罗拉多州布鲁姆菲尔德市

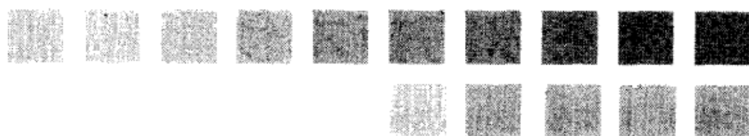
我的微波炉只需要有一个按钮，即“增加一分钟”。要烧开一杯水喝咖啡，我得按下按钮三次；要融化奶酪饼干，只需轻轻点击一下；为了加热玉米粉饼，我按“增加一分钟”，到点后过 15 秒钟再打开门。

只有一个按钮的微波炉会通过产品规划委员会的认可吗？可能不会。看看我的那台微波炉上的功能选项（都是从未用过的），我就知道，产品规划委员会喜欢复杂而不要简单。当然，他们可能会给“复杂性”冠以“功能丰富”的头衔。没有人从一开始就树立目标：生产的产品就得极尽复杂。复杂性总是意外出现的。

假设我有一块凉比萨饼需要热一下。根据制造商的说明，我应该按“菜单”按钮。我现在面临的选择是“速热”和“再热”。（嗯，我猜应该选择“再热”，虽然我有点饿了。速热会比再热快吗？）

“饮料”、“面条”、“比萨饼”、“盘菜”、“酱”还是“汤”？（我选择“比萨饼”，尽管它上面确实有酱并且它也放在盘中。）“熟食 / 新做”还是“冷冻”？（当然都不是，它只是吃剩下的外卖比萨饼。我猜应该选择“熟食 / 新做”。）“一块”、“两块”、“三块”还是“四块”？我不知道这样的盘问将持续多久，所以我按取消，然后按“增加一分钟”按钮。

这和软件开发有什么关系？就我而言，Amazon.com 只有一个按钮，即“一键购物”。哦，当然，我第一次访问时必须输入我的地址和我的信用卡号码，但现在我只要点一下就可以实现自己的冲动购物了。



软件通常要解决复杂的问题。现在的问题是，那种固有的复杂问题中有多少是你强加给最终用户的？你的软件是不是一个复杂问题放大器？成功的软件通常是一个复杂问题吸收器，因为它首当其冲地为用户承受了复杂的问题，而不是将这些问题一路传递给用户。

作为项目经理，你是复杂问题吸收器还是复杂问题放大器？最优秀的项目经理能够吸收程序员、最终用户和公司管理层等各方面抛出的复杂问题。当最终用户提出看似矛盾的需求时，你的任务就是帮助清理这些需求，而不是盲目地将这些问题传递给开发人员。当开发人员由于技术原因不能实现需求时，你的任务就是翻译（吸收）这个复杂的问题，并向用户进行解释，用足够多的信息帮助用户选择另一个途径。

使用你的应用程序有多容易？为应用程序添加一项新功能有多容易？对一项新功能提出要求有多容易？报告缺陷、部署新版本、回滚不良的版本又有多容易？

简单性不会偶然发生，它需要积极培育。而复杂性会因你不留意而丛生。



# 6. 偿还你的技术债



**布莱恩·斯莱滕（Brian Sletten）**
美国加利福尼亚州贝弗利希尔斯市

不管是普通的市民还是成功的企业，只要他们管理得当，债务就是一种有用的金融工具。它通过对未来利润的预支来平衡目前的资金不足。明智地使用短期债务能够有效地消除现金大涨大落。可一旦滥用，短期债务又会成为一个使企业举步维艰的沉重枷锁。

在软件开发界，为了实现那些“处于危险之中”的里程碑，同时完善大部分需要实现的工作，借用时间可谓是一种有效的策略。沃德·坎宁汉（Ward Cunningham）提出了“技术债”的概念，就是指在时间紧迫的情况下，开发人员为实现迭代[①]目标或快到截止日期时所采取的一种方法。在那一刻，他们无力用很恰当的方式来处理代码，但采取一些捷径，他们也许能够让编写好的程序“刚刚足以”冲过终点线。

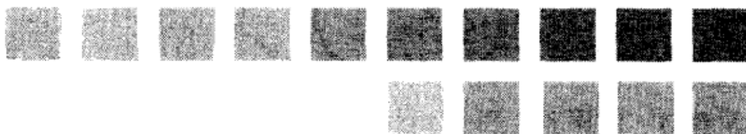
即使这套软件是临时的、有缺陷的，但只要有效地管理出现的技术债，这就是一种恰当的举措。可是，如果这笔债没有得到偿还，它就会变成一个令人痛苦的越滚越大的雪球。如果持续向未来借债而不偿还，那么就会令整个项目变得岌岌可危。

要想还清技术债，最好的方法就是评估在每个迭代结束前发生的所有“债务”。可以要求开发人员确定他们想要展开的临时修改[②]具体是什么，这样做需要花费多少时间。

---

① 迭代是指由一个敏捷项目小组选择的一段较短的时期（一周、两周或一个月等），在此期间，该小组会开发并测试由客户挑选的一个关键需求，并将结果发送给客户以征求其同意或意见。然后，小组会开始下一个迭代，以开发下一个最重要的需求并（或）纠正前一个迭代中完成的工作。

② 临时修改是指对一个在运行的编程问题作快速修复或解决，但这种解决方案并不是很理想，在时间允许的情况下可能需要重新修改。修改这种不完善的代码可以称为“展开修改”。



开发人员并不需要立刻还清债务，不过，趁他们对这些捷径仍然记忆犹新时来估量所需的修复工作，当然是不错的。

一定要发现要重写的代码有什么具体问题，而不只是随意判断所需要花费的时间。不要借机偷懒，这是一种严肃的、保持代码库干净的有效方法。

另外，有越来越多的软件分析工具能自动帮助识别出现“技术债”的地方，了解代码覆盖率、分析耦合性、检测风格偏离等，这些工具或许都可以在你不知情时工作。把留下“技术债”的地方记录到问题跟踪系统，并安排在以后的迭代中解决。只要可以平衡添加新业务功能的工作量并同时还清这些债务，就有可能既满足客户的功能要求又不使这些“技术债”失去控制。

软件难用的原因有很多，但通常都归结到临时修改、文档不足、依赖关系不恰当、快捷方式以及偏离预期设计等问题上。当开发人员最终举手投降说他们需要在一个系统上重打锣另开张时，实际情况极有可能是，许许多多没有偿还的技术债已经使他们不堪重负，债台高筑之下，他们觉得还不如痛快地宣布这套软件破产了。

只有一直坚持识别债务并迅速处理它，你才可能总是用“最低补偿”来避免接踵而至的混乱。在向业务利益相关者进行解释时，这种比喻说法可谓出奇地管用，能让他们立刻明白软件如何会随着时间的推移而变得无法控制，以及为什么应该投入力量保持代码洁净。

# 7. 为团队增添人才而非技能



**理查德·谢里登（Richard Sheridan）**
美国密歇根州安阿伯市

我过去采用的招聘标准同我们这个行业里的每一个人的招聘方式都是一样的：技能，技能，技能。直到有一天，一位来面试的应聘者当头泼我一盆冷水（这当然是一种比喻的说法），这次经历从此改变了我。

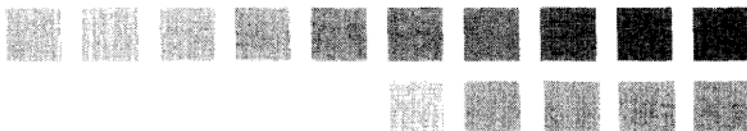
当时我正期望为团队招纳一个有多年微软工作经验的能人。仔细翻阅了比尔的简历，我觉得他非常适合这个职位。他在所有相关技术方面都有六年多的经验。如果我能开出不错的薪酬，这将会是一件轻而易举的事。

比尔接受了面试。我们先聊了聊，然后我向他介绍了我们手头上的一些项目，并对他说他是多么适合这个职位。我信心十足地认为这次招聘会进展得很顺利。可是突然间我意识到我会招不到他。我中途停止了面谈，问比尔发生了什么事。我告诉他，他非常适合这个职位，也对他直说我感觉到他不会来。

他的答复是："理查德，如果我还想做过去六年来一直做的事情，我就会留在我现在的单位了。我听说你们即将开始一些非常酷、非常新的Java项目，所以我才会想来这儿工作，因为我把它看成一个学习和成长的机会。"

就在那时我才恍然大悟。用"按技能挑简历"的方式来招聘新人，对一个要创建团队的经理而言，实在是最愚蠢的方法。

你知道，我和我的伙伴之所以会进入这个高新技术产业，正是因为我们希望走在科技的最前沿。我们没有一个人希望在职业生涯中重复使用自己在大学学到的那些技能。我们之所以进入这个行业，正是因为这个行业所涉及的永远都是新领域，永远都要学习新技能和新技术。



但不知何时出现了严重的错误。我意识到我们在员工成长上已停止了投资。我们不是在寻找新的人才，我们一直寻找的只是非常具体的精良技能。现在我要告诉大家，如果你们看到一个雇主在聘用新人时要求其技能达到精确匹配，那么，这个雇主其实是说："我们不打算投资你。"

对于所有努力创建强大团队的人，我要给出的忠告是：请记住你要聘请的是人才而非专业技能。当为敏捷开发团队招聘技术专家时，我要找的是什么样的人才？他们需要具备的优良技能不过是在幼儿园里就培养的：

- 能和他人融洽相处吗？
- 能主动合作吗？
- 当完成任务后，会把自己的工作整理以做备用吗？
- 会对新事物感到兴奋吗？
- 喜欢学习吗？

至于技能，我可以教他。事实上在我们的敏捷团队环境中，学习技术简单而迅速。然而，教一个成年人如何主动合作几乎是不可能的。

聘用人才而不是技能，这是组建团队的一种完全不同的方式。那些满腔热情、打算同我肩并肩一起开创激动人心的未来新技术的人，才是我想与之共事的人。

# 8. 西蒙，保持简单



**克利斯那·卡达利（Krishna Kadali）**
工科硕士
印度海得拉巴市康达巴

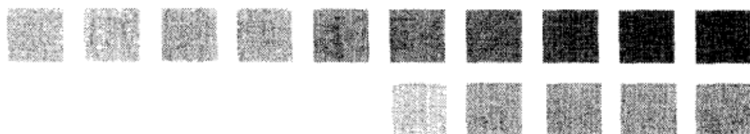
项目的利益相关者往往使事情变得过分复杂，这是软件项目失败的一种常见原因。项目组成员一定有能力完整构想该项目的目标并完成实际工作。可那些利益相关者却臆想，只靠几个简单的、不可思议的步骤就能完成这个项目。他们以为实现最终解决方案是轻而易举的一件事情，哪怕这个项目很复杂。

利益相关者不应该把软件项目建成一个统一、巨大而僵化的怪物，而应该让信息技术团队把项目建成一棵洋葱，每长一层就表示成熟一分。现实世界中没有其他可供选择的余地。不论那些需求有多完善，总是难免会有变动。如果你的软件不够灵活，不能迅速适应不断变化的需求，那么这个项目甚至在开始前就注定要失败了。

为了让事情保持简单，需要牢记下列关键点。

- **让需求保持简单**。需求分析人员往往会将自己脑海里冒出的某个解决方案同依据业务需要而提出的实际客户需求弄混淆。虽然实际的需求可能会非常简单，但是由于需求分析人员和开发人员之间缺乏真正的理解，会导致双方的交流并不到位。

  需求分析人员应该用简单树形图写出需求。根本需求是总体项目的简单目标。细小枝叶的子级需求被分门别类组成代表父级需求的枝叶。在整个图中不断重复这个过程，直到每项需求都清晰明了。可以使用软件思维导图工具来依照这种方法记录需求。只要明确了一个小子集的需求，开发工作就可以开始了。



- **遵循敏捷开发过程。**一旦确定了一个小子集的需求，开发小组就可以立刻开始创建原型。只要原型可用，利益相关者就可以测试并提供反馈信息。客户的反馈信息能够确保准确的需求，同时发现需求经需求分析员从实际客户传递到项目组时有可能形成的交流差异。让客户看到原型，也可以帮助检查开发人员设想的相应解决方案是否和客户所预想的一致。

  这些差异变成新的需求，于是开发人员重做原型，接着周而复始重复这个循环。每个循环周期应尽可能地短，通常不超过两三个星期。

  定义需求的一个小子集，按照陈述的需求创建一个原型，然后获得反馈信息，这种循环能够确保项目的所有利益相关者总能达成共识，且每个人对进展情况都感到很满意。只要认真地遵循这些简单的技巧，每一个软件项目都可以取得圆满成功。这里说的成功意味着顾客满意且软件实用，而且软件所提供的有效业务功能完全符合创建初衷。



# 9. 你并不是非比寻常的



**贾里德·理查森（Jared Richardson）**
美国北卡罗来纳州莫里斯维尔市

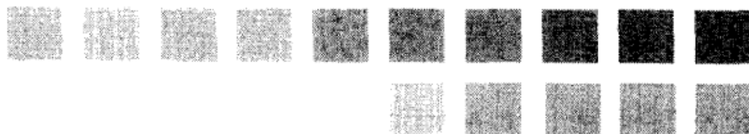
还记得你妈妈说的话吗？“你是非比寻常的！你是独一无二的！”其实，所有的妈妈都会这么跟孩子说。相信那个爱的谎言，结果导致了许多常见的软件项目问题。

我指导了许多不同的团队。当按照软件项目的达标情况进行考核时，那些相信他们是“非比寻常”的团队总是无一例外地落在了后面。因为他们自认为是非比寻常的，所以有着重塑一切的强烈倾向。他们认为：“其他的团队都不可能开发出实用的软件，或至少没有我们这个团队创造的软件这么优秀。”他们不从其他开发团队的错误中汲取经验教训，而是闷头重复犯着同样的错误。损失都由公司买单。

他们把大量时间用在重写、调试以及把自己的奇怪想法运用到那些已经成为行业标准的软件和工具[①]上，以至于他们永远都无法完成客户项目，而这些项目才是他们应该出售来赚钱的产品。这些虚幻而神奇的产品本可以像这个团队一样非比寻常，当然，前提是这个团队真的有幸将它们写出来。

听听这群独一无二的开发人员是怎么说的吧：任何现成的构建系统都不可能处理他们“独一无二”的需求。因此，他们必须为每个新项目编写一个新系统。不用现成的对象数据库映射工具，他们要写自己的。网络应用程序框架？他们信奉：“我们可以做。”持续集成？行。测试工具？我们也能写。他们当中那些最自负、最痴迷的人甚至还尝试自己编写编程语言。

① 工具：软件开发人员用来创建、调试、测试、分析、追踪或以其他方式支持高质量的软件开发的简单程序。



那么这些团队每天都在做什么呢？由于舍弃不用那些通常都可以免费获得的、功能完善的软件工具，他们要自己创建那些未经检验的代码，由此便出现许多问题，而他们每天就是在解决这些问题。如果是编写自己的数据库层，他们每天就要追踪隐藏的缺陷和问题。处理这些边缘事件[①]所花费的时间非常多，如果他们能学习或者修改现有的工具，那花费的时间将会少得多。

不那么“非比寻常”（但却更成功）的团队之所以会使用现成的工具，那是因为他们准备解决的问题都是很棘手的。他们需要可靠的工具，因此他们的注意力集中在软件项目的解决方案上，而不是试图为一个已经满满的工具箱再装入一个工具。

这和有效的软件项目管理有什么关系呢？不要让你的程序员白费力气做重复工作。如果他们跑来跟你解释他们的问题是多么非比寻常，你就直接向他们点明，他们的妈妈在做出“你是非比寻常”的评价时可能言过其实了。对可以利用的资源要做到了解透彻、信息灵通，引导团队使用高品质的开源或商业工具。

“非我发明不可”综合症使许多杰出团队脱离正轨。千万不要让你的团队也在此出轨了。

① 边缘事件：只在极端情况下（例如，最快或最慢的速度，最多或最少的数据量，或者最老或最新的浏览器接口）出现的问题或情况。通常是指将精力集中在琐碎且耗时的事情上，却忽略了重要的编程业务量。

# 10. 随时间滚动



**金·麦科马克（Kim MacCormack）**
美国弗吉尼亚州利斯堡市

12 年前，我的团队作为某家图形设计公司的分包商受邀开发一个网络应用程序。我们同客户没有直接的联系。客户将所有的需求都传递给总承包商，然后总承包商再通过一连串随意的电子邮件将这些信息转达给我们。

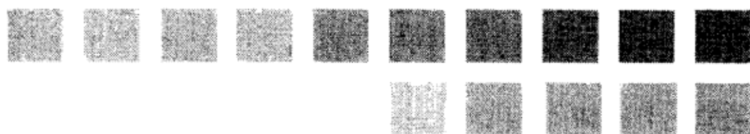
有一封电子邮件和我们的艺术家应该使用的屏幕分辨率有关。以前的标准一直都是 640×480，但最近的研究表明网站应该支持高达 800×600 的分辨率。（今天最常见的屏幕分辨率为 1024×768。）尽管这是一家经验丰富的设计公司，它却向客户陈述了这样的正式要求（我们从来没有见过）：

每一页的布局符合固定的 800 像素宽和 600 像素高的标准。

如果我们看到了这一要求，肯定会立即进行更正如下："每页的布局符合固定 800 像素宽的标准，而支持的显示器分辨率最高为 800×600。"我们做过许多网站，知道宽度是最重要的方面。用户讨厌水平地滚动网页，而垂直滚动则被看做是使用浏览器的一种现实情况，甚至是一种有利条件。然而，很显然，这条宝贵的真理从来没有被转达给客户。

这个网站客户为每个网页都提供了大量的内容。结果，只有极少数的网页能够在分辨率为 800×600 的 15 英寸显示器上被完整地纵向浏览。必须垂直滚动网页。

作为终端用户的客户并没有认识到，如果想在单个屏幕上显示这种超量的内容，我们必须是超人才能实现这个奇迹，于是他们非常失望。他们指责我们的主承包商，那家设计公司。反过来，这家设计公司便拒绝付钱给我们。据他们的说法，我们"没有满足书面要求"。



有了这次经验，我认识到粗制滥造的书面要求的危害性，也知道别人怎么利用它来对付你。永远都要用文档记录下你的设想，并坚持与最终用户而不仅仅是中间人来审查并签署这些需求，这一点相当重要。

幸运的是，敏捷项目管理实践缓解了这些问题。由于认识到开发人员和最终客户面对面接触的重要性，我们已经开始共同创建用户故事，开始根据他们提供给客户的商业价值而不是需求清单来排定各项功能的轻重缓急。一到两个星期的迭代过程意味着我们必须有早期和频繁的反馈意见，意味着我们有机会阐明客户的期望。

12 年后，我几乎遇到了一模一样的情形，一个客户希望在每个页面上显示大量内容，但同时也高度关注垂直滚动。幸运的是，按照我们如今的项目运行方式，再根据以往的经验教训，我们没有重蹈覆辙，而是迅速解决了这个问题，实现了客户的期望。



# 11. 你们的问题，我不买单



兰迪·卢米斯（Randy Loomis）
PMP
美国康涅狄格州安杜佛市

我们公司正在使用的培训软件已经滞后了五次升级，再不及时升级，供应商都不再提供支持。我们的项目就是与这家供应商进行合作，将培训软件升级到最新版本，然后培训我们的用户使用最新版本的软件。

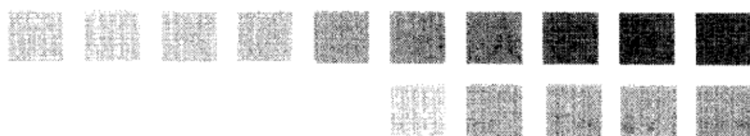
我们对工作作出了两项声明，一项概述了用户培训协议，另一项明确了培训软件升级的费用上限。在获得我们的数据后，供应商开始了远程开发和测试脚本[①]的过程，以便转换数据并应用第一次升级。

脚本一旦通过供应商的测试，就会迁移到我们的开发环境，我们要进行用户测试。在这五次接踵而至的升级过程中，每次都要重复这个过程。测试期间，我们会把遇到的所有问题都记录下来，待供应商重写、重测他们的源脚本后，我们再重新测试这些问题。

每次升级工作，用供应商的工作时间乘以声明中约定的费率，计算升级费用。进行升级时，我们会发现应用升级程序的错误，这些错误和我们用来安装升级写的客户脚本并不相关。我们会仔细记录每一个问题，打印屏幕界面，并提供一步步的细节，说明我们发现的是何种问题，也告知了我们是在哪里以及是如何发现每个问题的。

我们还要提供证明文件，说明供应商最早许诺过的软件功能。该供应商坚持认为软件是“按设计要求”运行的。后来我们发现，遇到的这些小 bug 只是冰山一角，它们还有更严重的后果。它们暴露出软件的基本功能存在严重的问题，甚至是在升级以后还存在问题。

① 脚本是指在计算机编程中由一个程序而不是计算机处理器执行的程序或指令序列。脚本可以用来控制软件应用而不用改变应用程序的核心代码。



过了一段时间，供应商承认我们发现的问题中有几个确实没有“按设计要求”运行，它们的确是 bug。由于约定了“费用上限”，虽然他们为了修改自己的产品，不得不做大量的工作，但是除了合同中约定的“上限”费用，他们并没有额外收取我们任何费用。

在项目的这个阶段，为了满足关键的最后期限要求，我们的注意力完全集中在软件的安装上。只能顾及一个问题到底是“按设计要求”运行的还是一个 bug。显而易见，如果我们当初明确记录了供应商修复每个 bug 所花的具体时间，那么我们可能都无需支付合同约定的上限费用。

当与供应商谈判合同时，应该明确约定双方（即供应商和你的项目团队）的时间要按遇到的具体问题进行追踪记录。这样一来，软件项目经理就能有准确的记录，如果问题与合同项目要解决的问题无关，而是出在供应商的原产品上时，软件项目经理就能够减少费用的支出。



# 12. 如何发现优秀的 IT 开发人员



**詹姆斯·格雷厄姆(James Graham)**
PMP
马耳他塔尔艾布安戈市

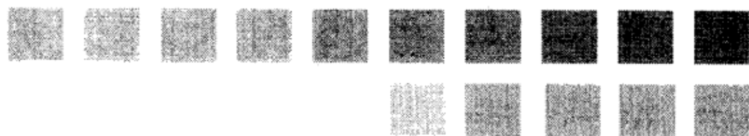
软件项目经理都知道，项目的成功取决于拥有出色的开发人员。你如何识别千里马呢？

面试新应聘者前，和最好的开发人员交谈一下。让他们重申一下所需要的具体知识。具有特定开发生命周期的经验、掌握具体方法或重要工具箱，以及拥有某方面领域知识（比如国防工业或制药行业），这些是开发人员最好具备的条件还是必须具备的条件？

要对其知识进行评估。你应和可信任的开发团队代表共同参与面试，还要附加理论测试。一个优秀的软件工程师能够立刻修复"模拟"的语法错误，并且不会精神紧张。他不需要看大量文档，也无需逐字阅读就可以看懂别人的代码，了解它的意图。当面对有问题的程序时，应聘者应该能够迅速找出问题，然后既能以"极客开发人员"的语言也能用非 IT 背景的利益相关者能听懂的语言描述它。

我们招聘编程技术人才时都认为其技能"越多越好"。但是我们如何界定"多"？尽管应聘者可能拥有丰富的知识，但是这个人可能还没有掌握有效应用它的技巧。在面对真实世界苛刻的项目时，一个刚毕业的大学生或刚培训过的开发人员，想要使用从课本上学到的理论知识时会很吃力。当最后期限一步步临近，所剩时间无几，而客户和其他利益相关者又施加了强大的压力，你除了有基础知识之外还需要足够的经验。

你和团队应该要求应聘的开发人员编写一段代码供你们审读。在分析了代码，并且与你信赖的开发人员讨论后，你才会知道这个人的方法和风格是否适合你的团队。



还要考察应聘者对待工作、同事、客户和利益相关者的态度。我曾经和一个被称为“吹风机”的开发人员共事。传说当他感到不满时，他就会用他的大声吼叫吹干人们的头发。他是一个优秀的开发人员，但是对于整个项目团队而言，他却是有害无益的。

编程世界正朝着敏捷开发方向发展，跨职能沟通和软技能将越来越重要。开发人员将会与公司中其他部门的人组成小团队一起工作。你未来的新队友若总是处于不受控制的自由状态，你跟他合作还会顺畅吗？招聘软件开发人员时需遵循以下简单指导原则。

- 审查他们是否掌握开发生命周期的正确知识、方法、工具，以及他们对所在行业（领域）的熟悉程度。
- 考察他们在工作环境下应用知识的能力。
- 测试他们的沟通能力和社交技巧。
- 寻找对工作有正确态度的人——既渴望创造出高端产品，又能接受项目的限制条件。是否有证据表明他们能及时且在预算之内生产出“切合意图”的产品？

不管你的应聘者多么有风度而且多么懂技术，都要始终核实发证机关的资格证书和前任雇主的履历条目。聘请阶段小心谨慎可以防止未来很多问题。

# 13. 优秀与普通的天壤之别



**尼尔·福特(Neal Ford)**
美国佐治亚州亚特兰大市

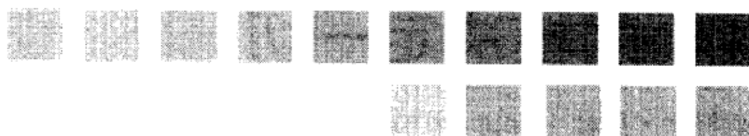
首次承担软件项目的项目经理在认识开发人员的技能方面容易犯错误，我们下面就要揭示这种错误。首先，要明白真正优秀的软件开发人员比普通开发人员要高效得多。事实上，统计数字表明，真正优秀的开发人员要比差的人员好上几个数量级，也就是成百上千倍。这里要强调的是，一个熟练的程序员不只是略高于平均水平，熟练与否的差异是巨大的。

新上任的软件项目经理做产品开发计划时，这个认识对他们意味着什么呢？经理们容易错误地认为，假使不能得到最好和最聪明的人才，普通开发人员也是可用的。但是开发软件并不像挖沟，在挖沟时即使最差的挖土机也能挖出一个洞。

在软件开发中，今天编写的程序会成为明天的基础。如果你让普通开发人员建设基础，那么为了继续前进，真正优秀的开发人员就不得不返工修复缺陷。雇用中等或普通的开发人员会减缓项目速度[①]。常常是，让一个差的开发人员离开团队比新加入一个优秀的人员作用还要大。

再联想到在延期项目中加人会使项目延期得更长，你就可以理解为什么大多数企业发展缓慢。没有经验的软件项目经理可能会争辩着说，因为增加更多的仓库人员会使货物装车更快，所以增聘程序员也可以缩短软件项目所需的时间。

① 速度：在敏捷软件开发中使用的一个术语，用以表明一个团队或一个团队成员在项目中的速率，即在给定时间内一个程序员能完成多少工作量。



这是行不通的。为了让新人跟上进度会花费时间，并且还会使其他程序员脱离既定任务。此外，沟通渠道会随着成员的增多而增多。对于一个只有两人的团队来说，只有一个渠道：贝齐·苏与比尔。增加迈克后，就会猛增到三个渠道。继续加人，渠道的数量还会继续以指数速度增长。

计算公式是 $n(n-1)/2$。在 12 人的团队中，你有 $12\times(12-1)/2$ 个渠道，或者说你作为一名项目经理必须维持 66 条关系。再增加一个人，就会变为 78 条沟通渠道需要监管。

用普通开发人员建设软件项目的想法体现了两个项目误区：1）你可以通过增加人员来缩短项目时间；2）普通开发人员以平均水平生产普通代码（bug 成群，偏离任务）也是可以的。实际上，普通开发人员会拖累整体生产力，并且使项目花费的时间比实际需要的更长。

怎样解决这种情况呢？一是给优秀的开发人员强有力的工具，你将更快获得高质量的软件。二是要认识到拥有普通开发人员对项目没有帮助，并且照顾差的开发人员会消减优秀开发人员（他们是工艺师）的生产力。软件开发太复杂，它不是一个装配线生产过程。

想要更快地开发软件怎么办？花额外的钱聘请和培养优秀的软件开发人员。这将在短期和长期（维护代码时）都给你带来回报。

# 14. 规模决定一切



**阿努潘·昆杜（Anupam Kundu）**
美国纽约州纽约市

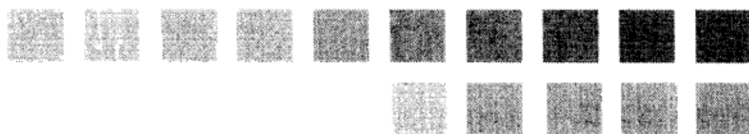
项目的规模、团队的规模、交付物的规模和清单的规模——项目中的一切做法都取决于它的规模。规模改变着游戏进行的规则。

项目经理应该把项目划分成易管理的模块，并与有才能的人分别承担这些模块，项目（在规模或复杂性方面）越大，这一点就越重要。这将确保关键项目成员（包括项目经理）在为项目做“体检”时可以看到全局而不至于迷失在细节中。

分布式项目往往比其他类型的项目规模大，因此，项目经理用来管理规模的策略实际上影响着该项目的底线。这个“大”字让人浮想联翩，它可能意味着 8 个人连续工作 12 个月（你是一个小供应商），也可能意味着数百人全年按合同做着维护工作（对你的客户而言你是一个 IT 巨头）。

这里有几个关于划定项目规模的建议，确保每个人都懂得自己那一小块可以导致整个大局的成败。

- 把项目尽可能地分成许多独立的但可掌控的工作流。
- 确保每个工作流至少有一个负责传送的关键联络点。
- 如果可能，尽量让主要成员在工作流中有交叉，以使整个团队可以共享大局观。
- 分别独立追踪每个工作流的进展（使用任何工具），并且定期监控节奏以掌控整个项目的动向。
- 分别记录和共享每个工作流的风险、问题、承担的责任以及依赖的条件。
- 定期举办小组会议，通报每一个工作流的状态。



- 公布整个项目的进度路线图，包括所有不同工作流的发布计划。
- 积极使用在线工具来共享用户要求、里程碑更新、错误报告、报告时限以及风险。

例如，假设你受托设计同一网站的三种不同形式（北美地区、亚太地区和中东地区）。你决定最好是创建三个不同的工作流，每个都有独立的传送联络人。由于三个站点基本上是不同版本的相同网站（相当于是中等规模的定制），所以最好让几个关键资源在三个工作流中自由移动。这样他们就可以确保几个网站的一致性，并可以重复使用具体实现细节。

另一个例子是一个项目可以有多个集成供应商的。比较理想的做法是将每个融合点（或相关的几个结合点）分离成单独的工作流，这样就能多渠道同时工作并且还能缩短传送时间。每日例会时让不同团队协调整体的传送质量。



# 15. 记录工作流程，然后严格执行



**蒙特・戴维斯（Monte Davis）**
MCSE
美国内布拉斯加州奥马哈市

我们试图把电子邮件从一个平台转移到另一个平台，但仅仅因为一个女人结了婚，我们的电子邮件系统便瘫痪了。

该电子邮件流程如下：

(1) 新的电子邮件通过新的电子邮件系统发送。

(2) 如果新的电子邮件系统可以发送，它就把信息发给正确的新系统用户。如果不可以发送的话，该邮件将被发送到旧的电子邮件系统来投递。

(3) 仍在旧系统中的人给仍在旧系统中的其他用户发送邮件时，邮件将会传送到对应的旧邮箱中。然而，如果收件人已被迁移到新系统，那么电子邮件会通过为每个用户创建的“迁移”转发地址转发过去。

有趣的事情出现了。一旦单身萨莉（Sally Single）被迁移到新的电子邮件系统中，她就有了两个电子邮件地址，一个是 sally.single@mycompany.com，另一个是转发邮件地址 sally.single@migrate.mycompany.com。所有旧系统的用户发送给她的电子邮件都会通过她的“迁移”转发地址自动转发到新的邮件系统。

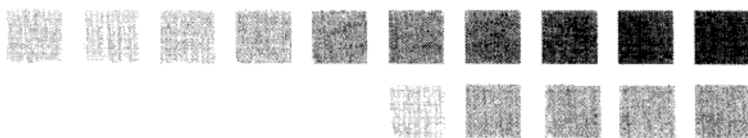
当萨莉结婚后，她把名字从“单身萨莉”改为“已婚萨莉”，她的电子邮件地址随之改变。然而，新系统中给萨莉的电邮地址重新命名的人忘记改变她在旧系统中电子邮件的“迁移”转发地址。所以萨莉就有了下列地址。

**新系统**

(1) sally.married@mycompany.com

(2) sally.single@mycompany.com

(3) sally.married@migrate.mycompany.com



**旧系统**

(1) sally.married@mycompany.com

(2) sally.single@mycompany.com

(3) sally.single@migrate.mycompany.com

（结婚后忘了改变这一条“迁移”转发地址。）

当旧的通信系统用户给萨莉发送邮件时，就产生了一个循环：1）创建信息并发送到旧的邮件系统地址 sally.single@mycompany.com；2）旧的邮件系统检查萨莉的账户，确保将邮件发送至转发地址 sally.single@migrate.mycompany.com，然后转发信息；3）新的邮件系统查找电子邮件地址为 sally.single@migrate.mycompany.com 的用户，但找不到它，因为该地址在萨莉结婚后改名了；4）新电子邮件系统转发了未知的收件人邮件到旧系统中；5）旧的系统知道用 @migrate.mycompany.com 地址转发所有信息，所以又将它转发到新的邮件系统中；6）如此以来，信息像泡沫一样堆积，循环往复地发来发去。

信息每循环一次，公司的免责声明就会被添加到消息后。虽然免责声明只有 100 字左右，但是当每个信息在系统中一分钟就循环几次时，内容就会迅速增加。由此显得萨莉很受欢迎。有这么多的信息发送给萨莉，其规模和容量终于导致电子邮件系统瘫痪。

这个故事的寓意：应记录工作流程，并确保严格执行。在这个故事里，虽然更名的流程已经记录好，但没有得到执行。否则，萨莉在旧的邮件系统中的用户账号就会随着她结婚后的新迁移邮件地址一同更新，问题就不会出现。

# 16. 剔除多余的流程



**纳雷什·贾因（Naresh Jain）**
印度孟买市马来德

成功的团队做了什么其他人都没有做的事情？他们不断质疑自己的流程，并努力消除多余的环节。他们毫不留情地对自己的软件开发进程进行重构。

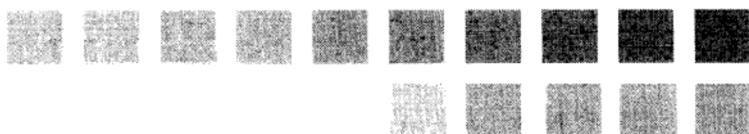
Il semble que la perfection soit atteinte non quand il n' y a plus rien à ajouter, mais quand il n'y a plus rien à retrancher。这个法国谚语引自安托万·德·圣·修伯里，意思是："完美，不是指无可增，而是指无可减。"

为什么今天的团队不应用这个准则呢？为什么随着时光流逝，最终产品的价值越来越少，而整个过程和副产品却越来越笨重？为什么代码行数在增加，但软件有用的功能却越来越少？

以下关键指标显示出软件开发进程出现了问题：

- 软件大幅膨胀，出现大量代码行和无用的功能；
- 开发软件的队伍的规模在不断扩大；
- 这个过程变得越来越规范化、教条化、僵硬化；
- 该团队正在经历"规划致死"会议；
- 文件和辅助产品总量以指数方式增长；
- 新发现的漏洞不断从客户测试组中涌现。

团队领导们不断增加更多流程、更多检查和更多审查，他们认为日益严格的进程将解决这个问题。根据我的经验，这绝对不是一个工作进程问题。添加更多的流程只会使团队更难看清问题的真正根源。



为什么大多数团队都不敢丢弃那些不能真正帮助他们的流程呢？为什么团队在一开始就用大量流程，只要能想到的都加上，而不是仅在需要时增加？

这可能反映出团队在最初使用这一流程时并不理解它。或者可能意味着，有人没有充分了解软件开发过程就给团队套上管理枷锁。无论是哪种情况，该项目都成了不切实际的计划，只能变成一堆无用的代码位。在没有明白造成项目在膨胀却没有增加价值的真正原因时，盲目改变是毫无用处的。

在我看来，一个好的项目经理应该正确理解团队的工作方式。他应该能够旁观者清，正确评估每个流程给功能性软件生产带来的影响。

一个经验丰富的项目经理应该仔细查看团队需要去做的所有活动，只保留那些对具体项目成功至关重要的活动。一旦将过去项目中多余的流程去掉，团队的生产力和工作量就应该很快能提高。

谈到流程时，“少即是多”是一个非常重要的哲理。



# 17. 矛盾体的需求说明书



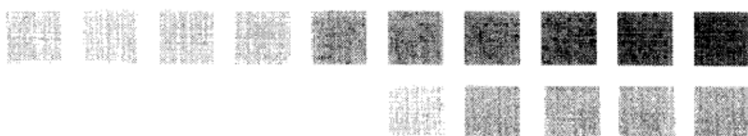
**艾伦·格林布拉特（Alan Greenblatt）**
美国马萨诸塞州萨德伯里市

**良好的需求**（R）描述一个产品的特性如何解决现有的或潜在的问题。**良好的特性**（F），也称为功能，被添加到产品中用以处理那些重要的问题。需求是由销售人员采集或者是由软件项目经理创建出来的。

- 我们想要在美国之外销售这个产品（R）。我们需要提供国际化和本土化支持（F）。
- 为了完成一项非常简单的任务，用户必须点击五个按钮。他们感到失望并且不会去完成任务。我们需要简化用户界面（R），并且将按钮点击数减少到两次以下来完成相同的任务（F）。

另一方面，说明书（S）具体描述了将怎样解决问题和满足需求。使用上面这些例子，下面的说明书或许会由系统架构师来写。

- 我们将抽出所有字符串，包括弹出消息，并且把它们放置在外部资源包里（S）。
- 这项应用将会被改进，在屏幕上显示的所有文本可以从那些资源包里取出（S）。
- 创建当地需要的特别资源包来实现本地化（S）。
- 通过点击按钮 1,2,3 才能完成的功能将会集成到单独一个按钮 A 上（S）。
- 现存的按 4 和 5 的功能将被集成到按钮 B 上（S）。



倘若不分清需求和说明书的界线，就会导致错误的人在作决定。导致的结局无外乎两种，要么是让软件开发人员来决定什么功能对客户重要，要么是软件项目经理来告知开发人员如何构建代码。无论如何，都只会产出低劣的软件产品。

开发人员没有经常和客户、用户、市场、销售人员和潜在合作伙伴交流，却试图了解什么功能是最重要的。另一方面，软件项目经理常常不是熟练的开发人员，他们不懂如何最好地去实现功能，不知道他们出于一片好心而提出的不专业的说明建议会给产品的其他方面带来什么影响。每个团队都有自己一套独特的技术专长。

良好的需求具体描述了你正努力解决的问题，并描述了为何一定要解决这个特别问题，这会使程序员在开发过程中更灵活、更高效和更积极。编码人员只有致力于解决问题并深入了解这个问题时，才会作出独立的设计选择。他们应该只受限于他们已经选用的技术，而不受制于非编程人员制作的教条式的脆弱说明书。

仍需要有需求说明书，但是它们可以应需而变。只有到了产品开发循环的末期，你才能完全理解一开始应该如何创建产品。

将你努力制作的东西与怎样去制作它区分开来。然后，让训练有素的各个团队成员根据他们自己在项目中的角色来做决定。

# 18. 商业价值始终是衡量成功的标准



**芭比·戴维斯（Barbee Davis）**
文学硕士，PHR，PMP
美国内布拉斯加州奥马哈市

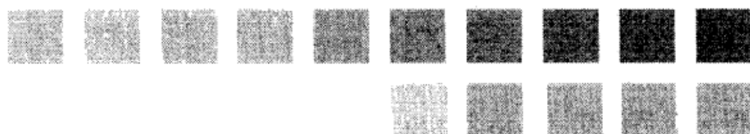
作为项目经理，很容易过分注重是否超出了时间、成本、范围和质量的规定。项目本身很快会结束，而我们的个人价值就在于我们是否有能力让这个项目实现上述这些可测量的期望值。

我们需要注意到一个事实，即项目是在获得它附加到组织中的商业价值时才是成功的。如果我们正在为市场开发一个软件产品，“成功”的评价因素是很明确的。我们需要使用项目管理技巧使产品更快地推入市场，那样我们才能在竞争对手生产出类似的甚至更好的产品之前让它赢得大量顾客。

我们需要在需求枯竭之前将产品销售给市场中的大部分顾客。我们需要将这个软件设计得使顾客易于安装和使用。它还要易于维护和更新。

许多软件项目经理感到他们的工作仅仅是完成软件。从组织的投资回报（ROI）观点来看，如果没有把项目和商业需求联系在一起，再好的软件也可能一败涂地。

如果是一个内部项目，那么它怎样才能为组织省钱或赚钱？我们开发的东西更快、更精简或者有更好的体系结构，是否我们需要的硬件资源可以减少？或是因为我们能够更快地接受订单、处理订单并且更快地送货，所以我们会赚更多的钱？还是由于我们开发的软件需要较少维护人员，或推出的基础架构可以降低帮助台的呼叫次数，所以能省钱呢？



如果我们的软件项目是为特定行业系统的集成商做的，那么，我们排序各种任务或者利用手头资源的方式是不是可以增加公司利润，或者赢得客户信誉？我们在项目上展现的非凡才能是不是可以确保我们能更快地完成更多的项目，从而得到供应商更大的硬件购买折扣？

当我们深刻认识到这个项目的完成可以给公司带来收益时，激励团队并且现场做出艰难选择就会容易得多。当在所有项目中选择主攻某一特定项目时，必须考虑为什么它比那些被搁置的项目更重要。

通常没有人主动将这些关键问题的答案告诉项目经理，所以你必须学会问。这些问题将提醒你思考，时间、资金或质量是否是项目的关键驱动因素。当你知道答案时，你就可以找出变通的解决方案，并且知道应急储备用在什么地方才可以让你的项目始终符合开发它的商业动机。



# 19. 不要总因项目放弃休假



**乔·泽尼维奇（Joe Zenevitch）**
美国纽约州纽约市

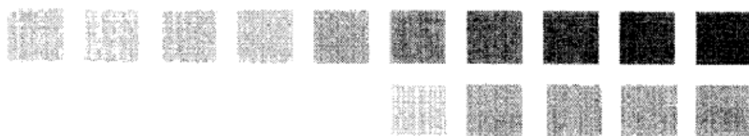
软件项目管理是一个要求很高的职业。除了在团队中位置显眼外，你还通常是个绝无仅有的角色，没有人能替代。想要休假是很困难的，特别是当你是一个第三方顾问时。你感到你的离开会对项目不利。

为了减少这种风险，项目管理新手常常取消他们的休假，或者更糟糕，根本没有打算休假。多年来，我已经认识到定期休假是十分必要的，它可以使你从大多数项目都固有的压力中得以喘息。在你年复一年的职业生涯里，你会多么深切地怀念那些错过的假期，但是致使你无法休假的具体工作的记忆却会变得模糊不清。

我不是建议你为了即将到来的假期而不顾你的项目或者项目计划。如果它只是一个三周的项目，你可以等待。在一个大型发布会前一周去休假很显然是不负责任的。但是如果一个 9 ～ 12 个月的项目因你在中期休一两周假而出现问题，那么你可能一开始就没有管好它。

显而易见，找一个合适的人并训练他偶尔顶替你是很重要的。他可能行事风格不会完全像你，但是作为你的代理人，他会保持项目之船平稳并朝着正确的方向航行。如果你喜欢将重要的事情推迟至你回来后再做，你可以让大家知道。

你的接替者可能是来自另一个团队的项目经理，但是从你自己团队里挑人更好。本团队成员将会非常熟悉这个项目，并且这给了他一个尝试担任领导者角色的机会。在一些组织中，业务分析人员可能非常适合作为一个临时的“你”。这个人熟知需求，如果你始终与各方利益相关者有联系，那么他应该会知道你的开发迭代周期的管理模式。



在敏捷开发中，让团队具有自我指导能力是很重要的，其作用十分强大。项目经理把易于理解且高度透明的工作流程放在显眼的位置。随着时间的流逝，团队采用且适应了这些程序，项目经理就会管理得越来越少，并且越来越顺利。从本质上讲，消除障碍和解决问题就会逐渐取代日常的细节管理。

顺利运行、自我指导的团队带来的令人高兴的作用之一，就是对于项目经理来说，按计划休假不是什么大事。就算你一连有几天不工作，这个机器仍然自我运转良好。

当然，要安排好你的假期，别耽误了出席项目发布会，但一定要抽出时间休假。永远不要仅仅因为没有你项目就会停下来而取消休假。



# 20. 集中精力



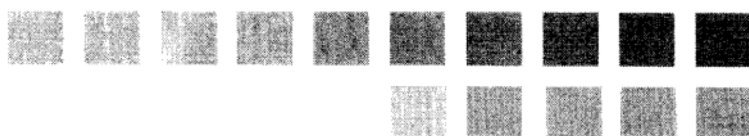
**詹姆斯·利（James Leigh）**
加拿大安大略省多伦多市

软件开发人员经常声称，诸如会议、演示和紧急改错之类的琐事常常打扰他们工作，使他们不能完成任务。通常情况下，一个人在经历一件琐事后需花费20分钟才能回到原来的思路上。一个仅需五分钟就能解决的问题实际上却用了25分钟，短短10分钟的会议会占用30分钟工作时间。各种琐事和重返工作状态所花的时间实际上消耗了一个脑力工作者一天28%的时间，并且还会引起过多的挫败感和压力。

为了解决这个问题，你可以每天留出两个小时（比如早上10点到中午）应对琐事。你也可以划出某一整天出来，在这一天里不允许安排会议、答疑、邮件、电话和其他让人分心的琐事，让开发人员集中精力做他们的工作。英特尔和IBM公司就留出了星期五，分别称为“零邮件星期五”和“思考星期五”。

同样重要的是，开发人员要知道他们需要优先考虑的头两项事情是什么，这样他们才能在限期内有效地安排自己的工作。在没有被明确告知什么才能带给项目真正的商业价值时，就算领悟力最好的开发人员也只能随意地瞎猜，抓不住重点。

信息癖（信息过量致人虚弱的状态）被普遍认为是影响开发人员生产效率的劲敌。编程需要开发人员同时想着许多东西——如变量、类结构、API、实用方法，甚至是目录结构。当一个开发人员思路被打断时，许多这类信息就从他的头脑中“换出去”了，此后还需要相当多的脑力才能回想起来。这会严重影响生产效率，研究发现，正是信息癖致使雇员不能以正常的水平来创造新的想法。



另外，不同类型的琐事对开发人员的思路有不同程度的影响。一个人可能在起身、上洗手间、喝咖啡、吃三明治甚至走动到白板前时仍然能够想着所有程序资料。实际上，这类活动可能会帮他找到一条解决问题的新途径。

计划中的会议对编程者来说特别成问题，因为他们提前知道日程表上有一个即将要办的事情时，就可能会浪费时间。他们会想："既然30分钟后要被打断，现在为什么还要开始呢？"并且开发人员在开会期间涌现出的好主意，会在返回电脑之前就被忘记，或者不再琢磨。

对于开发人员来说，他并行参与的项目每增加一个就会使他的生产率降低50%以上。同时参与三四个项目的开发人员经常需要花费更多的时间参加会议解释为什么工作没有进展，而不是在实际工作上花费更多时间。当开发人员必须投身于多个项目时，一定要确保在转到另一个项目之前，至少能将连续两天时间花费在上一个项目上。这样他们重新熟悉每个项目的时间总量就会最少。



# 21. 项目管理即问题管理



**洛林·昂格尔（Lorin Unger）**
美国新泽西州霍博肯市

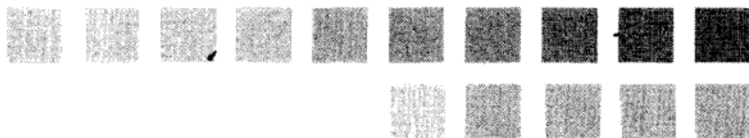
就算是在最好的情况下，软件项目管理也是极具挑战性和相当复杂的。然而，我常看见项目经理对自身角色有错误的期望而使项目管理变得更加艰难。

毫无疑问，项目管理就是问题管理。否则，就不需要项目经理了。如果一声令下，所有方面（资源、技术、需求以及时间表等）就能结合起来使工作很顺利地完成的话，那就不需要任何指导了。

但事实是，上述假设是不存在的，所以我们项目经理的角色才会存在。真实的情况是资源过度配置、技术和技能不匹配、需求不明确并且时间表不切实际。我常和一些项目经理共事，他们把这些问题看成是妨碍他们工作的麻烦、讨厌的东西或者是由外部力量造成的“问题”。“如果他们早已做好，或者他们深思熟虑，或是他们能给我更多的时间，那么所有这些不必要的复杂状况将会消失，并且我最终会顺利完成项目管理的任务。”

不必说，这些人大部分时间都会感到挫败、紧张、烦燥。

事实是，理顺所有这些不必要的碰撞和复杂状况是项目管理的任务。我们的任务是更好地计划、更清晰地思考，让项目资助者和完成它的工作者有更远大的战略眼光。之所以需要我们，就是因为执行项目本身就是一件容易造成混乱的事件，需要用我们多样的技能和个性去确保不可避免的困难得到化解、规避或者使其大事化小小事化了。



雪上加霜的是，这种情况不只是发生在管理项目上，有时候人也同样需要被“大事化小小事化了”。一个项目最具挑战性的方面不一定是技术或时间表，而是参与项目的人。这会是从负责项目资源配置到高级监督委员会中的任何一个人。

有这么一些典型的称谓：“气愤的资源”指那些总是不通情理地威胁项目经理权威的人；“紧张的利益相关者”总是不断焦虑而不能舒缓自己的情绪；“后座上的项目经理”，是指一有机会就会对项目运行指手画脚的利益相关者或项目参与者。

当然，讨论如何最好地管理项目中存在的各类人际关系问题超出了本话题的范畴。只能说在这个领域是经常需要处理这些问题的，并且这种需求在我们的项目管理职责范畴里，就像了解工作分工或维持精确的项目计划一样重要。

如果我们不把这些情形看成是完成工作的障碍，而是更恰当地认为它们是工作本身的核心，那么工作将会更顺利、更安宁和更平稳。当然，这是相对而言的。



# 22. 授权：蒂姆的故事



**肯·赛普（Ken Sipe）**
美国密苏里州圣查尔斯市

往往一个软件项目经理的最佳工作方法就是设定愿景，确定好优先级，然后就闪开。下面是一位名叫蒂姆的先生的真实故事。

我们发现项目需要增加一个小组成员，于是就发布招聘信息并着手面试。一个人很快成为我们候选名单中的首位人选。

他名叫蒂姆。蒂姆在所有的求职者中鹤立鸡群，傻瓜都知道应雇用他。但是有一票不同意。所有受雇到我们部门的人员需要在三个项目经理手下轮流工作。其中有个项目经理和蒂姆共事过，指出他缺乏主动性。她称蒂姆经常在工作时上网冲浪，像个懒鬼一样。

这真是一个棘手的境况。当作出一个雇用的决定时，除了看正常的面试之外，通常更要看重项目经理对应聘者的个人经验。不过从技术角度看，蒂姆的技能明显超过其他应聘者，因此尽管有一票反对，他仍被雇用了。项目正在使用一种敏捷开发技术，所以我们在迭代周期[①]开始前召开了一个公开会议。

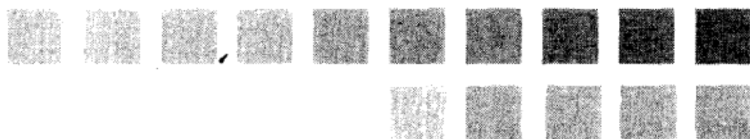
公开会议主要有以下几方面的内容。

(1) 根据用户输入创建故事[②]，确定它们的优先级并且交换意见。
(2) 通过这些故事创建一个项目范围的团队前景，建立良好的验收标准。

---

① 迭代周期：一个敏捷项目组选择的一段短期时间（一周、两周或一个月等），在这期间，开发、测试用户选择的关键需求，然后把它提交给用户试用或评判。然后下一个迭代周期开始，开发下一个最重要的需求和（或）修改在现行迭代周期中所做的工作。

② 故事：对软件需求的高级描述，常常分解为单个的开发人员任务，用刚刚足够的信息让开发人员评估他开发、测试和（或）实现它所需的时间。



(3) 要完成任务的开发人员把故事分解成多个任务并且评估它。

开完会后，任务进入了任务追踪系统。任务追踪系统的重要意义常常未被充分认识。它用来让开发人员看清什么任务已经开始了。如果他们早早完成了自己的任务，他们就可"偷取"（完成）一项原来已指定开发人员但还没有开始的任务继续工作。

事后证明蒂姆是一个优秀的雇员。在团队中他完成的任务比其他任何人都多。衡量他的价值的最明显的标准是他从其他开发人员那里"偷取"的任务数。完成的任务越多意味着项目完成得越快。

问题来了，为什么那个项目经理会认为蒂姆毫无产出呢？仔细观察就会发现，这位蒂姆的抨击者有一个过分控制的管理模式。她填鸭式地给开发人员布置任务，然后离开团队工作区去参加各种会议。蒂姆手快，会立刻完成指定任务。在没有人告诉他项目优先级或者他接下来去完成什么任务时，他就只好闲呆着。

当一个优秀团队拥有清晰的愿景、明确的验收标准和众所周知的项目优先级，而不是由项目经理一个人掌握、记录并且管理整个团队时，你将会为这个优秀团队的能力惊讶。有时候项目经理最好是无为而治。你会用好蒂姆吗？

# 23. 聪明代码很难维护



**戴维·伍德（David Wood）**
美国弗吉尼亚州弗雷德里克斯堡市

人们常要求软件开发人员创造奇迹。他们必须找寻聪明的方法使今天的项目代码能配合一身补丁的陈旧的遗留软件。通过发挥技巧和聪明才智，终于创造出许多行最终完成工作的聪明代码。但是由于所谓的聪明代码长且复杂，它可能只会制造出未来的维护问题。应该还有更好的方法。

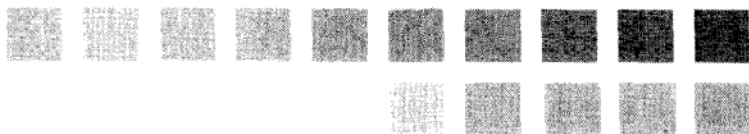
如果你是从事软件开发工作的项目管理新手，最好大胆地让开发人员探索更多的新语言和开发工具。赋予他们这种自由，因为这是他们发现新方法以改善编码实践和结果的必由之路。他们可能会对遗留的接口问题设计一个软件解决方案，不仅快速而且要测试和维护的代码行也少。这当然会成为你项目的一个优势。

相比你现在使用的语言，新编程语言用更少的代码行就能实现相同的功能。得到一个容易测试、能自定义、小存储和容易维护的简单代码结构，当然是很有价值的。

很明显，在组织中增加新语言和计算机平台会引发不少顾虑。这种新代码真的会解决目前软件开发或者升级的问题吗？它能与现在遗留（未升级）数据库使用的软件、组织中的用户界面和公司已投资的第三方软件长期融合吗？

还有，其他团队或部门的开发人员能用这种语言或平台开发软件吗？语言创建者能提供足够的产品支持吗？有及时的更新和升级吗？

即使你自己不熟悉编程，也要毫不犹豫地让程序员着手尝试新的语言。如果新语言能推本溯源回到 C 或者 Java（或者其他常用的做事方法），那么把它融入到你目前的实践工作中也不会很痛苦。



不管怎样，还是要确保在你的代码内记录下来任何新的实践，以免你的代码和关于代码的说明书不一致，那时理解系统的最好方法就只能是查看代码本身。这种情况称为在软件组件和系统元数据之间“失去耦合”。若没有足够的文档记录来维护你的软件时，就只能替换它了。

要鼓励你的软件项目团队开发人员创新，但是不要鼓励他们聪明过头，化简为繁。聪明过头会让其他人跟不上。如果后来的开发人员不能读懂代码，怎能期望他们维护它？任何程序员为了保住工作，都可能试着聪明地解决问题，但是没有项目经理能从中受益。

太过聪明的代码最终会难以维护。那会导致维护失败，对软件系统展开代价高昂的返工。



# 24. 掌控人的因素



**詹姆斯·格雷厄姆（James Graham）**
PMP
马耳他塔尔艾布安戈市

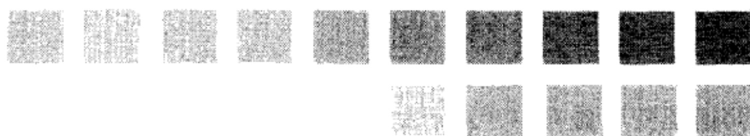
作为项目经理，我们常被具体计划细节所困扰。我们与队员聚集在一起，试着预测那些可能使我们项目脱轨的危险因素。我们捣鼓数字来确保能在规定预算内完成项目产出。但是我们总是没注意到或者完全忽视了项目失败最普遍的原因：人为因素。

从错误到事故，到彻底行不通，种种人为失败都可归咎于我们喜欢重复过去的行为方法。我们以为，如果上一次这么做成功了，那它应该再一次成功。有一个古老的谚语："对锤子来说，每个问题都是一个钉子。"心理学实验表明，在压力影响下，人们可能会回到像锤子那样的经验上，因为他们发现以前那么做成功过。还有什么比承接一个新软件项目更具压力的呢？

因为大多数项目的目标是开发新的产品、服务或解决方案，所以应该鼓励敏捷与灵活的头脑和工作方式，而非重复过去的行为。当你面对一个新的艰巨任务时，依照旧程序恐怕达不到原来的目的。

考虑在某正规软件项目管理方法上有丰富经验的业务分析员。从逻辑上，他可能会同意他的开发人员选用一个更敏捷的软件开发方法的理由，但是当面对危机四伏的时间压力时，他可能会考虑使用一些过去见效的惯用技术以及和软件无关的经验。

银行监管机构的报告指出，弄错数字是一个普遍的错误，特别是当雇员因为工作压力或个人压力而没有百分之百地集中精力时。知道这种人为趋势，明智的项目经理将仔细核定估算、预算等文件，重点寻找这几类易人为产生的疏忽。



是什么在给你的项目团队施压？它可能是个人私事，比如上班前和丈夫或妻子大吵了一架，或家里的财务紧张，或是家里有病人或孩子需要挂牵。

压力也可能与工作有关。它可能是一些小事，如与关键利益相关者开会时却迟到，忘记拿一个重要文件等等。或者，也可能是担心失去工作，害怕项目编码和测试不能按时完成。

压力导致人们沿用过去的行为而不是积极采取解决问题的行动。作为项目经理，你有责任找出会导致项目组成员回到过去老路上的压力是什么。通过与他们积极对话并且仔细管理好他们的工作环境，你可以为他们免除压力或者将压力的影响降至最小。

人都有情感，所以在工作中受情感影响是自然的。但是只有人才能开发软件，所以要像监管和保护你的其他资源一样仔细地培养和管理你的人力资源。



# 25. 使用维基

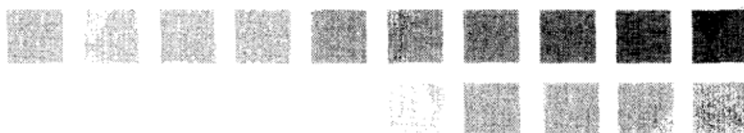


**阿德里安·威布尔（Adrian Wible）**
美国纽约州纽约市

维基是一个伟大的机制，可以聚拢你的项目信息。理想情况下，维基每天要更新多次，并且它在团队成员桌面窗口上总是打开着的。

为了防止浪费实际项目工作可能需要的宝贵脑细胞，我给出一些维基网页的建议。在创建这些网页的同时，你要知道怎样给自己的软件项目定制网站。

- **利益相关者**。为诸如最新的项目统计、短期问题、长期问题、风险、预算跟踪和里程碑成就等议题留下空间。
- **开发人员**。增加信息，比如连接到质量评价数据库的链接。伙伴们不会浪费时间去从其他随意资源中查找代码。团队人员可以共享议题，比如编码标准、构建和布署过程、常见陷阱和先进编码技术的使用（如依赖注入）。
- **一般信息**。软件项目经理应该在一般信息中增添帮助台电话号码、团队角色与职责，以及每个团队成员的联络信息。
- **团队日历**。使用本网站张贴团队日历。有个诀窍是使用一个指向 Google 日历的嵌入式的调用。
- **会议纪要**。将会议纪要存档，这样团队就能容易地想起过去会议的细节。团队成员也能迅速地查阅会议记录来研究问题，或为以后的会议作准备。
- **会议议程**。为利益相关者设立一个可以提示议程事项的在线程序。当然，经过软件项目经理的批准后，要将这一事项的必要性和下次会议的时间限制告知整个团队。
- **业务分析员**。这个人通常和开发人员不在一个地方工作。他需要能够进入工作文件和从多个位置访问领域制品的集中入口。



- **测试人员**。有些组织可能将测试的职责从程序员身上分离出来。该站点能为这两个团队提供情报交换所。应该贴出文章告诉大家怎样使用 Selenium、QTP 和 Quality Center 之类的测试工具。这里能在线开发和讨论错误跟踪程序，还能张贴最终的决定。

下面给出一些建议。

- 不要复制信息。如果信息放置在其他地方，可以链接它而不是把它复制到维基里。
- 注意变化的数量，确保信息没有过时。否则，人们将停止使用维基。
- 如果可能，试着让你的信息是由实时数据驱动。使用的项目管理工具要包含维基接口，图表和状态的创建由实际项目数据驱动。这将使工作进程处于实时状态。

任何时候你想通过电子邮件传送项目信息，尤其在有附件（文档、项目计划和预算信息等）的情况下，你都应该考虑团队维基是否更适合用来交换和存档信息。



# 26. 缺失的环节



**保罗·瓦戈纳（Paul Waggoner）**
MBA，PMP，MCSE，CHP，CHSS
美国爱荷华州沃基市

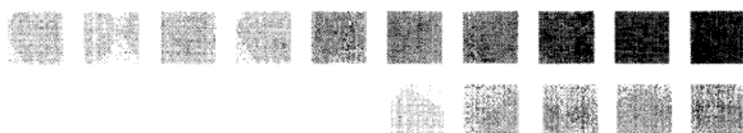
软件项目经理一致认为，他们最大的难题就是既要让团队成员恰当地处理项目细节，又要让他们掌控好分配的任务和进度。他们知道，团队成员总是面临分身乏术的两难处境，既要履行例行职责，如处理日常工作、解决问题、协调部门合作事宜、答复日常交流，又要按时完成项目开发任务。

被一个项目团队挑中可能在开始时会被视为是值得祝贺的事，但许多开发人员发现参与项目团队不可避免地要从日常工作中分心。到了紧要关头，开发人员甚至会公开承认日常维护和支持指定系统对他而言要比履行项目任务更为重要。

作为项目经理，你的第一个冲动是断定，因为项目工作现在或者将来都不具有优先权，那么这个人就不该属于你这个团队。然而，由于大多数组织的主题专家（SME）人数有限，所以可能无法再更换团队成员或寻找一个更敬业的人。

这里有一些小建议来帮助解决这个问题。

- 确保所有管理层支持该项目的目标。
- 修改主题专家工作描述，描述中要包含“作为团队成员完成其他项目中必要的工作”而不是“履行其他规定的职责”。
- 让管理层和人力资源部门强调这个改变，并且确保所有的主管在未来业绩评估中侧重考虑项目活动成绩。
- 在每个新项目开始时，软件项目经理、部门经理或主管、资助者或另一位关键利益相关者应该联名给每位项目成员发送一封个性化邀请信。这封信或电子邮件应该解释即将开工的项目的高层目标和每个具体团队成员的高层职责。



- 宣告项目成功完成时，要给每个项目成员颁发成就证书。要注意，这份证书的副本将放进他的人力资源档案中，用作季度业绩评估时的参考。
- 部门经理的上司应该明确一点，那就是与例行的技术任务相比，项目进展对实现组织目标更重要。
- 主管人员应该特别要求部门经理帮助项目团队成员腾出时间定期完成项目活动，不惜让他暂时卸下常规任务，交由 IT 团队中的其他成员承担。

应该明确认识到成功参与项目的人将使组织百尺竿头，更进一步，应该予以奖赏和表彰。对成功参与大小项目的人应该提出表扬。正如敏捷开发里常说的，这使得“可能的艺术”得以发扬光大，使员工的主动性与组织目标不谋而合。



# 27. 估算，估算，再估算



**理查德・谢里登（Richard Sheridan）**
美国密歇根州安阿伯市

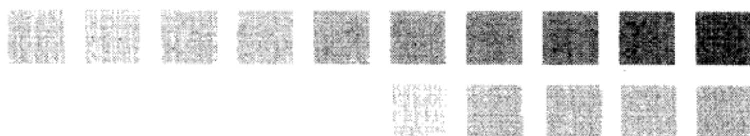
有种现象在项目管理中是如此常见，我们在项目初始阶段（当我们只知道一点点时）就对项目进行估算，然后在整个项目进程中（比刚开始知道得多些时）却对以前的估算结果不闻不问。更坏的是，我们从来没有把最后的实际结果和我们最初的估算作个比较，没有为我们的将来锤炼技能。

在我们的组织实践中，我们要对每个项目一周估算一次。甚至是先前已经估算过但还没有实施的任务，我们仍然要再次估算。为什么我们要这样做呢？有以下几个原因。

- 我们估算得越多，就会估算得越准。
- 有时候，我们逐渐知道得更多，这能帮助我们进行估算。
- 有时候，我们逐渐意识到不像我们原先想的了解那么多，这也能帮助我们估算。
- 通常，在涉及新技术时，早期的估算会包含"恐惧"因素。随着对新技术了解得更多，基于恐惧的成分会逐渐减少。
- 估算是一项团队活动，它是我们一种重要的交谈方式。

最终，要想能更好地进行估算，最好的方法是确保随时了解实际情况，这样团队就能得到估算结果的反馈信息。我唯一要提醒的是：不能用这个信息来惩罚团队！估算的真正作用不是让人们达到他们的估算目标，而是一旦他们认为达不到时，可以事先提醒你。

这里有一个你能玩的简单游戏，可以让你充分理解估算和反馈的力量。拿三个大小不同的空瓶子，用吉利豆装满。记下装满每个瓶子各需要多少吉利豆。



召集一个你正给其讲解估算的小组，让每个成员估计最小的瓶子里面有多少吉利豆。我会让人们两两配对来估计。

给他们一小段时间提出一个估计，然后让他们写下来。让每组大声读出他们的估计。在一个白板或书写挂纸板上写下估算结果。对第二个和第三个瓶子重复相同的步骤。

最后，告诉团队成员这是一个做估算的好方法，感谢他们的参与，并且在你离开前问是否还有问题。我屡试不爽：一定会有人问每个瓶子里到底有多少吉利豆。他们想要知道！小吊一下他们的胃口，然后告诉他们这问题有多么可笑。不过是一瓶吉利豆而已，至于吗？

现在切入正题。问问他们有多少次在更重要的题目上应该把信息反馈给他们的团队，却认为不重要而没有做。从此后，他们再也不会疏于给他们的团队反馈信息了。



# 28. 项目管理办公室在前进



**安杰洛·瓦尔（Angelo Valle）**
巴西里约热内卢市

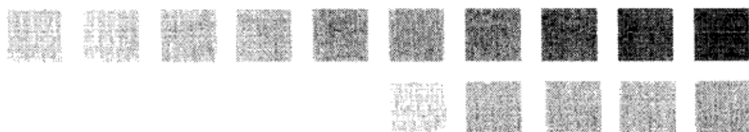
如果你是一位软件开发人员，你可能认同一个反应灵敏、适应性强的机制才能开发成功的软件。遗憾的是，似乎全世界的所有人都在所有部门推行标准化的方式。这唯独对软件开发人员来说不是个好消息。

组织结构中最新的做法是成立项目管理办公室（PMO）。这一遍及全球的现象是让一个小团队承担起监督和支持企业项目和程序的任务。团队的目的是把一致性引入文件和模板中，让报告程序标准化，并且提供一种统一的方式通过项目增加企业价值。

PMO 的目的是成为智慧和协作的核心。通过实行统一的组合管理、计划管理和项目管理，他们在部门项目中将企业战略目标同员工的行动联系在一起。这对保障工作质量而言也是件好事。

在企业内部，PMO 的职能如下。

- **战略职能**。在这个职能中，PMO 要鉴别和挑选项目并给项目排出优先级，这些项目与公司的战略计划是高度一致的。
- **指导职能**。为了履行他们的指导职责，PMO 的职员确定指导方针、标准和模板。他们评估和指导软件项目经理应该如何应用最好的方法、工具、技术以及软件来成功完成开发团队的目标。
- **支持职能**。PMO 为项目团队成员和项目经理提供支持。包括开办培训课程，调整模板和文件以使其适用于所有部门，或者与项目经理一起解决人员配置或其他人力资源问题。



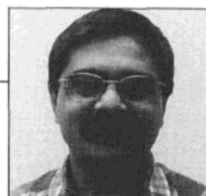
世界各地不同公司的PMO都不相同。每个公司的项目管理工作也处于不同的发展阶段。所以通常意义上的PMO可能包含各式各样的职责，有一些如上所述，也有一些本文未提及的独特职责。

PMO指导大家选用合适的经验证的标准化工具、技术和软件，因此可以减少由于不确定性和日益强调项目应更便宜、更好或更快而引发的问题。PMO在确有必要和行之有效的地方使用了一整套标准化的方法学：项目鉴别、数据收集、分析、信息采集、发布、报告、风险管理、采购、质量和其他项目管理知识领域，如文档记录和通信。

在国际会议、研讨会和最近发表的论文中，通过PMO模型而取得经济上的成功已成为热门话题。如今，大学教育越来越需要适应当前“真实世界”的实践，因而，关于这个话题的学术讨论也很多。今天的学生就是明天的开发人员。

PMO已为多数人所认可。如果现在你正好是一位软件开发人员，你应该主动开始和PMO积极讨论，交流一下专业人士成功的经验，谈谈软件开发流程的独特性。如果不这么做，你极有可能会被各类不适合的方法、文档需求和规程所拖累。又快又好、质量又高的软件对每个人都是最有利的。

# 29. 重苦劳，更重功劳



**温卡特·苏布拉马尼亚姆（Venkat Subramaniam）**
美国科罗拉多州布鲁姆菲尔德市

软件开发需要花费很多精力。然而，如果你听到有人自夸“我写的这个应用程序有 300 多万行代码”，那么你就要问问他这个程序里到底有多少行代码是真正需要的。

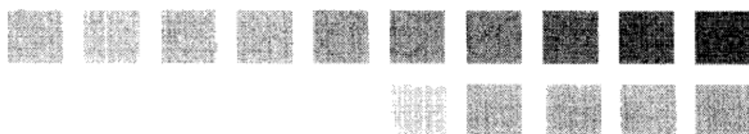
添加额外代码通常是考虑到某种扩展性[①]。扩展性很重要，但是如果没有做好，它会起到反作用，从而延缓你目前的项目。

此外，超范围代码表明软件项目经理只看重超时和超力。如果你墨守成规地要求程序员必须加班加点地工作，那么一定要确保他们工作能产生额外且可用的结果。

我喜欢草坪总是绿茵茵的，于是便让洒水系统每天自动为其浇水。那是我在科罗拉多州度过的第一个夏天，我注意到一棵枫树的叶子落了一大半。我猜想这是高温和干旱引起的，所以我延长了浇水的时间，但是没有起色。我咨询的专家问我：“你浇水的频率和时间是多少？”我如实回答后，他说：“那就是问题所在！将浇水时间和频率减半，就会有起色了。”

我一直在用过量的水扼杀那棵树。少浇一些水果真起到了作用。这样做增强了这些树的抵抗力，帮助它们茁壮成长。两周后，我的树就恢复了健康，枝繁叶茂起来。

① 扩展性：一种把未来增长考虑进去的系统设计原则。在编写当前所需功能时，尽量考虑创建和执行额外功能的能力。



谈到工作时间，你的程序员就像枫树。占用他们少量但充足的时间，指定一些宽泛的任务，他们就会茂盛起来。给他们布置艰巨的任务，要求他们经常加班，他们就会开始枯萎。另外，因为他们工作时间太多，所以往往会写多余代码，把程序搞复杂了。

我过去的一位上司就相当在意员工工作时间的长短。周六上午是不是在上班，或者是不是一直加班到深夜，对他来说，这比员工的实际产出还重要。连续工作 12 小时甚至一整天的程序员不可能多产而高效。

在另一个团队中，经理要求我们保持传统的八小时工作制。是的，有些天我们会走得晚一些，但是这些是例外而不是常规。员工知道他们不需要长时间工作，但是必须尽职尽责按计划提交工作结果。所以我们精力很集中，很少分心，也能很好地分清工作的轻重缓急，并且高效利用时间。尽管两个团队开发人员的能力不相上下，尽管我们在第一个团队里工作到筋疲力尽，但我们在第二个团队所取得的成绩更多。

鼓励程序员报告他们所取得的进展，而不是报告他们工作多长时间。让他们知道你关心他们取得的成果，而不是想掌握他们在电脑前花费的时间。一旦团队成员意识到你是一个注重成果的经理而不是一个“投入时间”的经理，他们的关注点将转到获取结果而非数着钟点工作。

# 30. 软件的失败是组织的失败



**布莱恩·斯莱滕（Brian Sletten）**
美国加利福尼亚州贝弗利希尔斯市

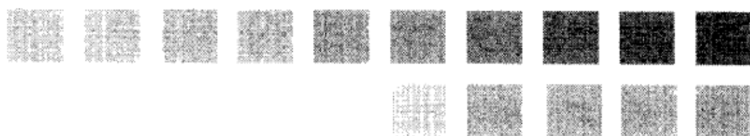
当组织中的软件项目出现问题时，我们总是责备开发人员。当错过最后期限时，或者当交付的产品中的 bug 比一个昆虫学家的异想天开还要多时，似乎表明这个团队不够好、不够聪明、不够有成效、不能应对挑战。尽管一些团队可能确实应该受到一定的指责，但是你不能忘记，成功的软件项目需要所有利益相关者积极充分地参与。

每个人的参与是至关重要的。为避免失败，你需要知道谁何时、为何正在创建什么。增添业务功能时你需要慎重并且要分清轻重缓急。你需要发现那些需求表达不清的问题。你需要认出谁会阻碍进步，发现沟通不力的地方，安抚被压垮的（但仍渴望成功的）开发团队，从而将潜在的障碍扼杀在萌芽中。

开发软件需要有效的度量标准、清楚的沟通交流以及全身心投入的业务和管理人员。他们必须参与交付软件成果的努力中，并且，不论结果是好是坏，他们都应承担一定的责任。软件项目经理需要测评和记录成功和失败的经历。一贯能完成交付的团队是值得信任的。很少能完成交付的团队需要更多的监管、进一步的训练和重新组合，说不定一些成员还必须被请出团队。

尽管如此，还是给软件开发团队一些时间去自己清理乱象。当他们匆匆赶着完成各种产品发布[①]时，他们将遭遇维基先锋沃德·坎宁汉（Ward Cunningham）所谓的“技术债”。像真实债务一样，如果没有坚持不懈、认真履行还债义务，这笔债会变得繁重不堪，花再多的精力也无法维护。

① 发布：敏捷软件开发方法在多个较短的时间框架内分别创建一些具体功能。在每个时间段内，都要执行需求分析、计划、设计、编码、单元测试和验收测试。在这段时间末期，可用的功能就得以“发布”，即在顾客面前展示。



工作的每次迭代[①]应该包括新的业务功能，也包括对一些不可避免出现在代码中的修改进行重构[②]。这既不是说允许不认真工作，也不是表明这个团队不好，它仅是一种编程的现实，而相关的管理者必须给予一贯的全力支持。

组织必须努力跟踪行业发展趋势，获取恰当的工具，并采纳优良实践提升程序员的工作效率。要鼓励开发人员在工作时间内外拓宽他们的知识。尝试新工具、接受培训、参加高价值会议、阅读书籍和博客，这些都是在这一领域必需持续努力去做的。

组织团队午餐让成员共享知识并推广新理念，这种方式对促进发展而言既有效又实惠。软件开发工程师一旦感受到来自老板的支持，往往就会表现得更加忠诚，更愿意完成额外的工作。而且，他们更有能力随时应对需求与技术风格的改变。

软件业中有很多工作可以帮助从业者一贯按时发布高质量的产品。软件开发组织必须全方位地投入生产过程，以促成自己连续不断的成功。

① 迭代：是指由一个敏捷项目小组选择的一段较短的时间（一周、两周或一个月等），在此期间，该小组会开发并测试由客户挑选的一个关键需求，并将结果发送给客户以征求其同意或意见。然后，小组会开始下一个迭代，开发下一个最重要的需求和（或）纠正前一个迭代中所做的工作。

② 重构修改：重新编写一个快速可行的修改程序来让一个软件功能工作，但这需要更进一步的内部改进以方便其长期使用和支持。

# 31. 来自另一端的声音



**马蒂·斯科莫（Marty Skomal）**
MPA
美国内布拉斯加州奥马哈市

倾听开发人员和软件项目经理的声音，这当然不错，不过，你可能会发现，倾听拿着钱包的那个家伙的声音，对你同样有帮助。我是顾客。

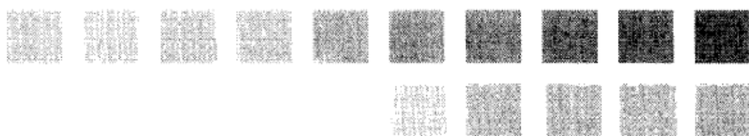
软件开发人员现在已经渗透到非营利性组织和政府部门，许诺以低成本、基于网络的方式来经营业务，并使用在此以前一直费用高昂、太过复杂、让我们员工和选民感到不舒适的奇特技术。

靠小本经营维生的非营利性组织和政府部门，会被这种美好、自动化的未来所诱惑，但是这里有一个陷阱。试图拥有一切，你最终会一事无成，会非常怀念过去用鞋盒、用一套 3×5 的索引卡片来存放数据的日子。

比如，我所在的部门决定将基于纸质文件的审批流程转变成网上审批。需要填写的表格直接发到我们部门并下载到数据库中，这避免了人工数据录入错误，降低了成本，并避免了发邮件给选民带来的不便。我们也能在他们准备时在线审查申请，在其提交前提供协助。

我们的软件开发人员迫不及待地指出他们如何能使审批流程的其他方面自动化，比如，在允许组织进入系统前审查他们是否符合资格标准，确保不错过截止日期，确认报表收支平衡才能点击提交按钮。

我们的核心需求只是导入数据、检验其准确性并且通过电子邮件回复说我们已收到他们的申请。然而，他们鼓励将我们的系统编制成了截止日期之后提交的申请会被拒绝。由于设立了严格的条件，我们失去了反应敏捷、以服务为导向的灵活性。另外，一旦系统在截止日期后阻止了一个申请，如果不去联络开发人员特意重写记录，我们根本就没有办法把它输入到数据库中。



我们本应该从一个更简单的系统开始，在熟悉了其功能之后再增加其复杂性。可是恰恰相反，我们本需要一辆完整的自行车，结果却得到了一个不完整、不实用的宇宙飞船。

我们摒弃了那个系统，现在使用一个功能较少但更稳定的系统。我们调整自己的内部程序来适应这套系统，而不是从头创建软件以完全保留我们过去的流程。现在我们的网上审批系统所做的就是在自己的数据库里接收并处理数据，它不再是一个华而不实的纪念碑，不再只展现那些技术上可行但不一定实用的功能。

为了避免让你的非营利组织客户误入歧途：

- 让他们稳扎稳打地计划和开发，并且测试，测试，再测试；
- 抵制诱惑，不要建议他们把简单的任务过分自动化；
- 要成为关心理解用户需求的开发团队。

请在新兴的市场尽情施展你的技术手段之前，先充分理解你的非营利组织客户想要成功地实施什么。

# 32. 保持洞察力



**詹姆斯·格雷厄姆（James Graham）**
PMP
马耳他塔尔艾布安戈市

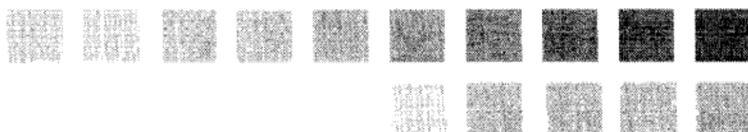
从用户那里收集业务需求时，通常听到“系统缓慢”、“应用程序不稳定并且会崩溃”、“我们需要的它不做，我们不需要的它却做”、“菜单结构繁琐”和“完成一项简单任务要点击太多次”。

大多数软件项目经理会深表同情。我们提出一些似乎能去除他们痛苦的解决方案，想让他们感觉好一点。虽然用心良苦，但我认为这种方法本质上是错的。更进一步说，它降低了新项目成功的可能性。

一些人说业务需求的要点是为了提供量身定做的解决方案，相应地减少最终用户的挫折感。我本该认可这是一个有价值的目标。然而，如果项目经理决定最佳解决方案的依据只是热切地想取悦用户，那将会导致严重的问题。事实上，项目经理可能还没有训练自己保持不偏不倚的洞察力。

洞察力意味着寻找最好的解决方案，而不是感觉适合用户的简单解决方法。记住，用户对他们的业务领域有着深刻的理解，如果能分享出这些知识，他们就能为项目作出巨大的贡献。但是我们应如何利用他们提供的信息呢？

当我在伦敦做管理顾问时，经验丰富的同事常常教导我要重视客观性。他们的智慧来自一个不言而喻的道理，即大多数专家都喜欢显摆自己有多聪明，可越是这个时候，他们往往越应该多花些时间，发挥自己的专长提出正确的疑问，从而揭露根本问题。如果没有揭示出真正的问题，则所有努力只是治标不治本。



我们都有犯这种错误的危险。前不久，我接到一项任务，为一家大型企业设计一个管理开发程序。我的第一反应是赶紧解决他们的痛处，让他们来看看我现成的一个程序。我知道，只要将这个程序简单改一改就能解决让客户恼羞成怒的问题。

幸好这时我的自制力开始发挥作用。我花了一个小时与对方的高级经理讨论真正困扰他们的问题。在冷静下来听取业务问题之后（不是仅仅听取最终用户抱怨可能存在问题的地方），我推荐了一种完全不同的解决方案。这套方案更适合他们的需要，能解决他们的核心问题。

当你下次面对沮丧的用户时，请先做做深呼吸。允许他们发泄自己的不满，讲述他们每天都要遇到的表面症状。这些恼怒当然是发自内心的真情实感。然后你再提出一系列的问题，去发掘引起他们沮丧的根本原因。千万要克制，不要为解他们的一时之痛而提供一种暂时的解决办法。在设计解决方案之前，确保瞄准的目标准确无误，这才符合客户的最佳利益。



# 33. 怎样定义“完成”



**布赖恩·萨姆-博登（Brian Sam-Bodden）**
美国亚利桑那州斯科茨代尔市

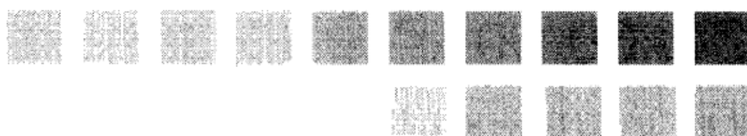
如果没有明确定义“成功”意味着什么，一个软件开发团队就很难取得成功。对开发人员而言，成功需要交付一个满足客户期望的产品。然而为定义整个项目的成功，我们需要在较大的项目团队中对“完成项目”给出一个公认的精确定义。

为了涵盖整个项目范围，传统迭代软件开发的核心原则是“分而治之”。项目被划分为多个可交付使用的产品，每个可交付产品又被分成多个工作包。每个工作包最终被划分成多项任务，分配给每个具体人员。

当使用一个只有一周或几周的迭代（工作周期）的敏捷方法时，可能不用去考虑整个项目范围。完成一次迭代的目标可以被清楚地定义为，创建的软件通过了单元测试，也许还通过了有限的集成测试，可以把预期的软件功能演示给利益相关者看，以获得他们的认可和反馈。

问题在于，从宏观层面来看，除了代码和相应的测试，一个项目还要考虑许多其他方面。如果使用传统瀑布式方法，测试总是沦落到项目最终阶段，这将是流程中的缺陷。如果使用较敏捷的方法，开发人员又可能会错误地推迟考虑或完全不考虑与编程无关的事项，因为他们看不到软件项目必需的其他方面。

在新创建的和先前迭代里创建的部件（功能）之间要做的单元和集成测试，就属于这些易被忽视的事项。



这些事项被忽视凸显了开发团队里的一个重要问题。软件的复杂程度似乎与相互联系的部件数量呈几何比例关系。一定要花时间精心制作演示环境、编写用户级 / 验收测试脚本和创建相关文档，做这些工作可不能漫不经心。不管你的方法如何简易，可交付的软件都需要一定数量的文档资料。

如果以上事项都得到了重视，你就知道“完成”的宏观定义完全不同于在一次迭代内累计完成多少个功能集。同时，每次迭代中由于完成那些忽视的事项而形成的增量，也会改变大家实现和测试功能的方式，以及客户对功能的理解。

我们不要因为行政或者商业的事情来加重开发人员的负担。我们需要传递的重点理念是迭代不仅仅是开发人员的事。对更大范围的软件项目团队成员来说比较重要的任务，都必须和这些迭代协调一致。业务分析人员、软件项目经理和质量检查人员必须和这些开发人员一起协调他们的关键活动。

软件项目经理负责这种协调，他必须从宏观层面上理解并传递“完成”的完整含义，以便与编程无关的事项能够与每周的迭代工作齐头并进。在明确“完成”的实际含义时，软件项目经理必须做开发团队和其他利益相关者之间的仲裁人。

# 34. 60/60 定律

**戴维·伍德（David Wood）**
美国弗吉尼亚州弗雷德里克斯堡市



我们常常自认为软件开发是软件生命周期中最重要的部分。围绕着开发人们提出了大量的方法。书籍、期刊文章和博客都在关注开发。然而，开发恰恰不是花费的重点所在。

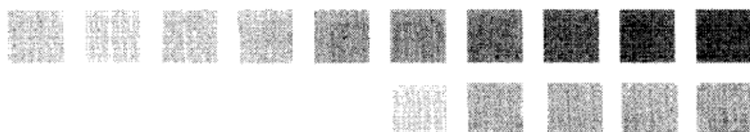
软件系统生命周期的总费用中有足足 60% 来自维护，只有不到 40% 来自开发。当然这只是平均数。根据系统类型及其应用的环境不同，实际的维护费用可能从 40% 到 80% 不等。在维护中，平均来说，60% 的费用与用户产生的改进（需求的变化）相关，23% 与迁移活动相关，17% 与修复 bug 相关。

生命周期中 60% 的费用与维护相关，而维护活动中 60% 的费用和改进相关，二者联系在一起，就给了我们所谓的 60/60 定律，这是人们提出的为数不多的几个与软件维护相关的“准则”之一。

迁移活动包括把系统移植到新硬件或软件环境中。当然迁移也是一种需求的变化。把它也估算进去就反映出一个有趣的事实：超过 80% 的维护活动都或多或少与需求变化相关。

当然，一个人有能力改变程序意味着他应该首先读懂它。维护期间的一项主要活动就是理解要作出的改变。大概总维护时间中的 30% 是花在对现有软件产品的理解上。不论是需求变化、迁移还是修复 bug，所有类型的维护都需要发挥理解力。

为了维护他人写的代码，或是为了维护很久以前我们自己写的已然陌生了的代码，理解它都是我们必须付出的代价。对大多数任务而言，维护期间对代码的理解就相当于开发期间设计新的程序。



60/60 定律应促使我们重新思考软件开发的核心以及维护工作的核心。目前这种专注于开发活动的趋势可能并不会产生最大收益。在 20 世纪 80 年代初波姆（Boehm）认为，恰当的软件工程准则能减少高达 75% 的缺陷，这可能会是真的（尽管我仍深表怀疑），它已成为许多开发方法的基础，但是那又怎样呢？

一种好的方法可能会减少 bug（占总体维护工作的 17%），但是根本没有处理迁移、优化的时间或者成本。为了减少维护成本，我们就得考虑理解代码、使代码适应新的需求和（或）迁移代码到新的环境所发生的成本。

60/60 定律表明我们应该集中精力创建可维护的系统。我们必须将软件设计成可变化的，这样，在面对新需求时，系统就可以灵活自如。在软件工程领域，设计这样的系统是我们即将面临的巨大挑战。

我们至少知道部分答案。软件各部件之间不需要那么紧密耦合，应该像万维网一样，各部件在最后必要时刻才以灵活的方式捆绑在一起。



# 35. 遭遇敌人……敌人就是我们自己

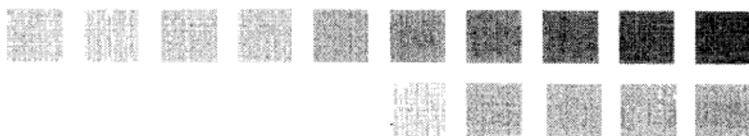


**芭比·戴维斯（Barbee Davis）**
文学硕士，PHR，PMP
美国内布拉斯加州奥马哈市

漫画家沃尔特·凯利（Walt Kelley）创造了经久不衰的连环漫画Pogo，他有一句名言经常被人引用："遭遇到敌人……敌人就是我们自己。"对软件开发流程感到陌生的软件项目经理最能准确体会这句话了。下面将介绍如何避免"敌人"就是你自己。

- 作为项目经理，你期望团队成员能够估算出他们完成一项具体任务所要花费的时间。如果他们超出预算时间太远，就会对整体进度造成不利影响。你对该项目所负的责任之一就是组织会议推动团队交流。你需要非常仔细地展示自己估算并交付会议的能力，就像你期望开发人员估算和交付他们的代码一样。

  如果会议时间太长，你就是在窃取开发人员宝贵的编程时间，而他们正指望用这些时间来完成项目的进度。
- 如果项目团队说外语，你就要学习这门外语并请一个翻译。你的开发人员不说你的语言。你应买一本书、上一堂课、让Google成为你的朋友，并且找一个擅长用简单方式解释复杂事物的开发人员。如果不了解这些概念、术语以及团队所面临的挑战，你就别指望这个项目能蒙混过关。
- 你过去从事的工作或许是制造烤面包机、生产汽车、制药或盖摩天大楼，无论以前用过的方法如何完美，在这里都将不起作用。让团队中可信任的成员为你解释敏捷方法吧。它们不是新事物，也没有风险。但是它们最有可能保证你在项目结束时取得可行的产品。



- 开发人员是工匠和艺术家。他们的工作不同于会计师、律师或者银行出纳。当他们三三两两聚在一起谈笑风生时，他们实际上就是在工作。当他们对着墙拍球或者在白板上写写画画时，他们可能正在精心构思一个架构问题的解决方案，而这个问题是不可能靠盯着电脑屏幕就能解决的。请给他们空间。
- 你的团队工作的时间不是固定的。我们都见过当地食品商店的收银员交接班时的情况：准备下班的收银员打开收款机，更换收款抽屉，然后，要接班的收银员便上来开始工作了。程序员不可能像这样和他的同事交换位置，直接从同事结束的地方开始工作。当你的团队成员正兴奋地工作时，不要去打扰他。研究人员说，如果一个人的工作被打断，他可能至少需要一个小时才能恢复先前的生产效率。
- 没有必要让每个人都用相同的语言来编程。有一些新语言只需要写很少的代码就可以解决问题，这样一来，需要测试、在服务器上存储和维护的代码都少了，不妨一试。不要拒绝让你的开发人员使用最适合工作的工具。

勇于接受软件开发的新世界，你就能成为软件开发团队的支持者而不是敌人。

# 36. 工作循环



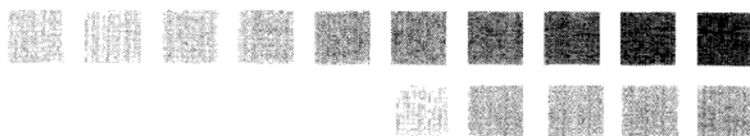
**詹姆斯·利（James Leigh）**
加拿大安大略省多伦多市

我们的身体有许多生理周期，我们的生产效率也受此影响。大脑集中精力持续关注一件事情的时间不会超过几小时。一天的工作时间要合理规划，确保身体有时间休息，并且每隔一个半到三个小时能重新精神饱满。事实证明，工作一个半到两个小时之后，生产率就会降低，这时就需要大脑转移注意力，才能保障生产效率。

生产环境要让开发人员的智力劳动高效，才能创造出最有效的软件项目。然而，有助于提高开发人员生产效率的许多事情都不在软件项目经理的掌控之下，比如，你不能确保他们吃得好或者晚上睡眠充足。尽管如此，项目经理仍然有办法保证开发人员的生产效率在白天不会下降，只要鼓励他们经常休息，并为他们提供补充营养的机会。俗话说得没错，开发人员是受他们的胃驱动的。

研究还发现，项目被分解成多个迭代时更容易成功。通过创建每周或每月的子项目（要完整地说明目标、优先权、反馈和发布），能减少 bug 并提高开发人员的满意度。把工作划分为更小的迭代，使大家有机会跟踪流程并认可好结果，也让团队中的每个人有机会思考别人的工作并提供反馈，从而增进同事之间的交流。

每次循环应该包含规划阶段、行动阶段、完成阶段和奖励阶段。在开始一项行动之前，先问问你自己或你的团队 5 个 W：为什么、何时、如何、什么和谁？为什么我们要做这件事？何时将会完成？如何做这件事？我们将实现什么？该项目的每个部分将由团队中的谁来负责？通过适当的沟通和理解，行动阶段会富有成效，这有助于整个项目的成功。



完成一项任务或行动项目后，听听外界的反馈。如果团队中的一两个成员完成了该项工作，就让另一个团队成员来重新审查它（同行评审）。如果这项工作是由整个团队完成的，就从其他的利益相关者那里获得反馈信息（最好是终端用户）。任何循环的最后阶段都是奖励阶段，对团队和个人的可持续健康成长来说，这是很重要的。如同电池在使用后必须充电一样，大脑和身体必须通过大家对工作的认可来获得犒赏。

作为软件项目经理，你必须领导团队完成一个个项目循环，确保每个人理解计划并得到他需要的反馈。此外，每个人每天都必须遵循他自己的循环，包括计划、工作、完成和获得奖励。项目经理必须确保所有项目成员得到关注、信息和时间，他们需要这些来保持自己的生产效率。通过这种方式你能确保团队正开足马力，尽其所能地工作。



# 37. 先照顾好自己



**哈里·塔克（Harry Tucker）**
美国新泽西州马塔瓦恩市

任何一位坐过商业飞机旅行的人都记得，飞机安全示范要求我们先戴上自己的氧气面罩，然后再协助年纪较大的人或者儿童戴上。如果在遇到危险情况时，我们努力先为别人而不是自己戴氧气面罩，说不定还没等到为他们戴好面罩，我们就会因为缺氧而死亡。先快速戴上自己的面罩，我们才会有能力使出 100% 的力气照顾别人，最后大家都活下来。

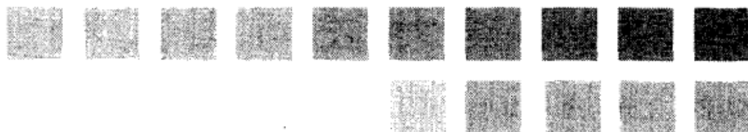
这些年来，我目睹了许多出色的项目最终都以失败告终，有时那完美的成功就近在眼前。这些存在问题的项目都拥有：

- 无限的市场潜力，
- 优秀的产品，
- 能力非凡的自主团队。

然而，由于项目经理失去了自我控制，因此，他也不能照顾整个团队。也就是说，良机因缺氧而消失了，最终项目也失败了。

为了管理或者领导团队（二者间有明显区别），软件项目经理需要完全控制自我。他们必须透彻理解自己的目的、愿景和使命，以及个人目标和职业目标。有了稳定的个人生活，经理才能掌控自我。如果失去这样的掌控，经理就会很容易被生活中遇到的挑战打倒，于是，一个聪明又有天赋的经理就无法将精力集中在手头的管理任务上。

这种情况要发生时，不同的征兆就开始出现了，一个接着一个。



(1) 经理明显变得心烦意乱，并且开始有失控的感觉。

(2) 身不由己，他觉得没有能力挺身而出去保护项目。

(3) 失去保护的团队开始经历沟通失败的痛苦。

(4) 沟通失败导致任务滑坡（达不到原来的项目标准）。

(5) 任务滑坡，经理没有能力恢复对事态的控制，导致团队陷入绝望。

(6) 团队士气下降，让一个本已失控的项目雪上加霜。

我自己会花时间进行每天、每周和每半年的回顾，审视自己处于什么样的生活状态。每天和每周的回顾会帮助我聚焦短期目标，每半年一次的回顾（就如一些人所称的空白计划）让我有机会评估我的长期目标，确保我在个人和职业两个方面都处于正轨。

虽然生活会一直给我们抛出难以击打的曲线球，但制定短期和长期目标会帮我们明确目的，在经历风雨之后重新回到正确的人生和职业轨道上。有了这样一个计划，我们就能够更加专注于现在的任务，包括管理我们的团队，提供他们动力，助他们走向成功。

氧气面罩已经掉下来了——你会先帮谁戴上？

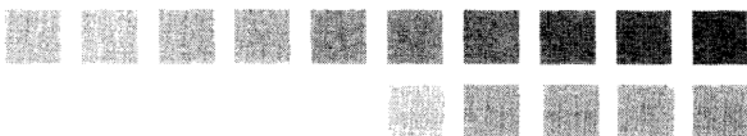
# 38. 别指望开会写出代码



**威廉·J.米尔斯（William J. Mills）**
美国加利福尼亚州卡斯特罗谷

在绝大多数情况下，那些常陷于会议中的人本可以将事情做得更富有成效——这种会议会使他们迷失预期目标，浪费时间，甚至会让整个团队都徒劳地坐到会议室里（而这个议题本来不需要那么多人就能解决）。因此，除非有明确目标，否则不要安排会议，而且就算要开会，也只邀请那些有必要参与的人们出席。作为一名软件项目经理，当你为团队安排组织会议时，千万不要出现下述情况。

- **闲聊**。如果有的与会人员打算利用这个时间就项目进行非正式的交流，你可以提醒他们在开会之前早来一点，或者在会议之后再安排时间一起讨论。你不能让整个团队都在等他们聊完。
- **未参与、未完成、未解决**。制定好一个清楚的议程表，并事先分发到相关人员手中。如果你要召集整个团队开会，那要确保会议主题与每个人都相关。
- **讨论过于深入**。在会议上提出已经显现的风险或障碍是好事，但会议上不是提出解决方案的地方。可以组成更小的团队或指定适合的团队成员在会后继续深入探讨这个问题。一旦软件工程师们开始谈论具体的实施细节，就要提醒他们将这个问题先做好笔记，等到会后再解决，你要继续进行议程表上的事项。



- **偏离话题**。当前的会议有具体目的。不要迷失焦点。会议上可能会出现一些很重要的问题，但是这些新问题不在讨论议题范围之内。因为就这个新问题来说，在这次会议上可能会有许多与之不相关的人在场，又或者有许多解决这个问题的关键人不在场。所以你要么安排另外一次会议，要么结束现在这个会议并开始处理新问题。记住，这时要让与议题无关的人回去工作。
- **花费太多时间**。作为项目经理，你期望团队成员能够评估他们完成某项具体任务所需花费的时间总量。如果他们在会议上花费的时间超过预算时间太多，这对计划也是不利的。
- **开会过于频繁**。如果你遵从敏捷方法，日常短会是必需的。但如果不遵从敏捷方法，那么一定只在必要时才召开会议来传达信息或搜集数据。
- **纵容与会人员长篇大论**。作为会议的领导者，示意“停”（朝发言者举手）并显露微笑也是你的职责。对他说：“你说的我们已经很清楚了，但是由于时间关系，我们需要进行下一项议题。”（或者，“听听别人的”，“如果有时间的话，我们再回到这个问题上”。）

因为你才是项目的领导者，所以你的团队成员不大可能就低下的会议效率畅所欲言。那么，这时你就应该审视一下自己的行为，看看是否能通过自己的努力来加以改善。

# 39. 绘制变化进程图



**凯西·麦克杜格尔（Kathy MacDougall）**
美国科罗拉多州伊利市

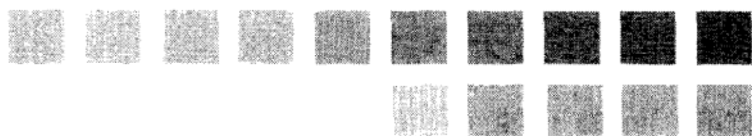
新软件会改变人们的工作方式。这对组织来说可能是好事，但是在组织中工作的人们并非时刻准备好去欣然接受这些变化。我们必须承认，如果不能说服、哄骗或命令人们使用你的新软件，那就是时间与金钱的极大浪费。

当要求人们改变他们的工作方式时，使用诱饵的方法（奖赏）比使用批评（惩罚）的方式要好得多。如果你的新软件为用户带来的显著好处是他们不能理解和掌握的，那么即使强迫他们采用，他们也会寻找种种可能的方法去避免使用。你应该适当留意的是：1) 明白这种改变对它所涉及的人们的影响；2) 计划并实施一些帮助人们能够欣然接受这种变化的管理计划。

要想理解变化对人们带来的影响，关键就是要理解人们目前是如何工作的，并弄清楚这个新软件将会如何改变人们现在的工作流程。这会提高用户采用新系统的可能性，也会完善最终产品的设计，因为这样做能确保新软件满足用户的需求。

不能低估变化管理的重要性，它应该是项目经理在项目早期的重要关注点。为了确定这种变化对用户的影响，首先要记录下目前（“真实的”）所有与该软件项目相关的流程。创建记录日常工作任务以及数据输入和输出的流程图表。

接下来，记录下一旦新软件推出后这些流程将会有什么不同。与新软件的目标用户坦诚地交谈。讨论这些变化将如何影响他们的工作。认真地听，评估每一个功能变化带来的影响和成本，并且对软件设计作出相应的调整。确保目标用户和他们的管理人员会接受这些变化。



及早并且经常联系目标用户群的经理。他们将是变化的重要支持者，通过激励和（或）要求最终用户进行改变，他们要么促成要么破坏向新系统的过渡。他们是为软件得以推广而清除障碍和解决未知问题的重要伙伴。

为变化制定一份计划。确定项目启动前需要进行的训练和团队建设活动，并且将这些加入到你的项目计划表中。争取来自管理团体的帮助，以创建变化管理计划。最低限度，这个团队必须全心全意地接受相关想法、实施和培训方法以及时间期限。如果有人认为某些方法可能不适合团队，就要仔细聆听这些反对意见或警告。

总起来说，用户和他们的管理人员始终关注的就是实现他们的商业目标。转变到新的流程、工具和系统对这些目标构成了潜在威胁。详实的前期计划有助于提供向新系统的平稳过渡，为认同和接受铺平道路，并且提高长期使用的可能性。



# 40. 达成共识



**戴维·迪亚兹·卡斯特罗（David Diaz Castillo）**
MBA，PMP
巴拿马巴拿马城

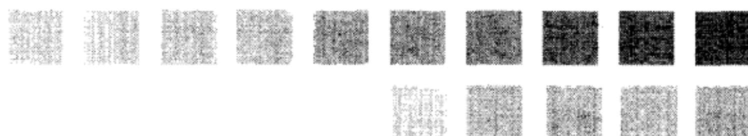
组织常常会把几个相关的技术项目进行组合，变成一个更大的项目计划。该策略是想以一种更加符合成本–效益优化的方式来完成这些项目，并避免这些项目之间的重复或脱节。然而，软件项目经理一定要把其中每个小团队必须完成的真实业务目标传达给项目团队成员，并且告诉他们如何与其他项目保持一致，这一点至关重要。团队成员也必须明白他们这项具体目标的完成对于整个计划的成败有多大贡献。

以下是一些关键任务。

- 寻找这些小项目之间的相关性，以全局的眼光考虑这些项目的价值。
- 确定计划中所有项目共有的关系、可交付物以及风险。
- 使所有团队成员与这个大计划努力要完成的最终解决方案保持一致。
- 深入了解业务，当不可避免的问题出现时，能够提出符合计划战略目标的解决方案。

将多个项目集合到一个大的计划方案中的概念在拉丁美洲还不是很成熟，但是我们正在尝试运用一套共同的管理政策来管理多个计划，并从中积累经验。让每个团队对于整个计划目标（而不仅仅是它自己承担的项目任务目标）都有清晰的认识，这无疑是大有益处的。

最困难的事情是获得外包商、客户、赞助商以及其他利益相关者的认同。我们需要分析他们的兴趣、要求以及需求，以确保多个技术项目的组合对他们每一个人都是有价值的。项目组合所获得的价值应该超过单一项目独立完成的价值。



特别是当客户范围跨越许多国家时，这些客户都会对如何创建技术有自己不同的观点，并且也会有自己独特的组织程序和流程。为了使所有利益相关者都与计划方案目标保持一致，首先一定要就多国认可的项目管理方式达成一致。

为了有一套成功的方法，我们需要通用且灵活的文档或模板，可以适用于所有项目。当我们管理 IT 项目时，通常我们都需要按照外包商各自的方式和模板从他们那里获得商品和服务，或者需要运行周期进行的其他项目的成果。所以在我们开始之前，各方都必须对我们将要选用的文档达成共识。

我们的项目管理方法将会是什么样的呢？如果每个小的项目团队不能就整个大团队将要采用的方法论、程序、流程和整体变化控制步骤达成共识，那么就很难实现这些小项目的共同发展，从而也无法给大计划提供更好的服务。进行选择时，软件项目经理需要询问团队成员，哪些模板和方法是合理的、实用的，并且能帮助他们有效执行和控制他们的项目。

一旦拥有了一个通用流程和文档（模板）工具，你就可以将技术项目组合到大计划中了。这些大计划将为你的客户和组织带来超过任何单一项目的价值。

# 41. 依据现实制定计划



**克雷格·莱特维克（Craig Letavec）**
PMP，PgMP，MSP
美国俄亥俄州韦恩斯维尔市

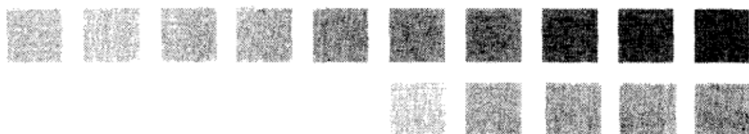
延期、超预算、质量不过关的软件项目多得让人感到不可思议。即使墙上挂满了国际资格认证和软件能力成熟度等级证书，软件公司在管理软件开发的动态环境时也还是会出现许多麻烦。

开发速度在项目生命周期的各个阶段自然是不同的。有时你赶在了计划之前，有时又会落后于计划。项目经理们通过严格、详细的项目时间表来努力掌控这些波动，时间表会详细描述任务的分配和最后期限。然而，项目经理们会发现，在他们疲于应对软件开发动态本性的过程中，他们自己却在对计划不停地进行修订。

尽管对于任何项目而言，要想成功，就一定得按照详细、精心估算的计划来执行，但许多软件项目经理可能会发现，在他们的计划中添加一些"现实时间"会大有益处。

关键链法使用"缓冲区"来处理项目生命周期中不可避免的各种变动。尝试着在软件项目开发周期的每个阶段（设计、编码和测试等）引入"缓冲时间"或"现实时间"。

"缓冲时间"能够在无需作出重大调整的前提下，为项目的每个阶段赋予一定程度的灵活性。可以把"缓冲时间"看做是一个阶段的应急储备时间。做法相当简单。看看项目里的每一个阶段，依据最佳计划考虑一下这个阶段需要花费的全部时间，然后在这个阶段的末期添加一个缓冲任务，这个缓冲任务所需的时间暂定为这整个阶段所需时间的 10% 左右。



例如，一个设计阶段需要 40 天完成，在这个阶段末期增加 4 天的缓冲时间，这样一来，这个阶段共需 44 天。这个阶段果真会需要 44 天吗？或许不会，但是这“没有使用”的时间可以进入下一个阶段或者添加到以后其他的缓冲时间里。

经验丰富的软件项目经理都知道，项目在早期阶段可能会按计划进行，但到了后期，都会变得拖拖拉拉。因此，走在时间曲线的前面几乎总是利大于弊的。

在你第一次尝试这种方法时，一定要做好准备迎接怀疑的目光。当领导审查你的计划时，“非生产性”时间是他们想要删除的第一事项。你一定要坚持自己的立场。直接告诉他们，你正在执行的是基本的进度风险管理。

如果在你的项目中，有一个阶段风险更大，那么就为那个阶段添加更多的缓冲时间。当然，你也可能在时间表上的其他地方添加较少的风险缓冲时间。

最后，确保你没有“重复添加”。“重复添加”是指已从任务角度增加了额外的时间，然后又从阶段角度增加额外的时间。最佳状态是，你已经习惯性地为每一项具体工作增加了额外时间来处理未知事项，而无需为每一项任务都增加缓冲时间。

试试吧，它很有效！

# 42. 完美实现的谬误



**戴维・伍德（David Wood）**
美国弗吉尼亚州弗雷德里克斯堡市

如果你认为工作足够努力就能创建完美的代码，不要不好意思。许多人也这样想过。遗憾的是，这是不可能的。甚至理论上这也是不可能的。

在一般情况下，很难检验任意逻辑，要想进行全面测试就更难了，甚至不可能。借用建筑领域的砖瓦来打个比方，英国的 3 名研究人员最近表明，软件是很难检验的，因为“没有好的、可预见的建造材料。构建程序的组件包括语句、例程和对象，这些都不可能以可预见的方式组合在一起”。

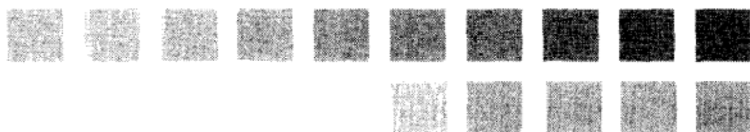
构建软件的组件不是像乐高积木那样扣合在一起。它们的组合方式多种多样，根本无法确定所有的组合。这种情况可能恰恰印证了“图灵完备”[①]。总而言之，软件是非常复杂的。

跟踪和验证代码中的任意逻辑听起来很深奥。简单说，就是追踪程序员的意图。当然，我们可以与程序员讨论，询问他们的意图。遗憾的是，程序员通常会在写完代码数天后就忘记了自己的思路，特别是当需求发生变化或者记录的文档前后不一致时。

程序员也会改变原定工作，留下没有文档的或者文档有误的代码。源代码迅速成为探寻程序员意图的最后一项、也是唯一一项法庭证据。哎！只能通过类似于变量名、逻辑流和偶尔的注释等线索来模糊确定他们的意图了。

---

① 图灵完备（Turing completeness）：以阿兰・图灵（Alan Turing）命名，是指计算机设备的每一种设计都可以通过通用机来模拟。真正的图灵完备机在物理上是不可能的，因为它们需要无限的存储能力。不过，假如那些机器具备无限存储能力，它们就成为了通用机，也就实现了图灵完备。



每一个发布的软件产品中都不可避免地有 bug。软件中出现 bug，既有严重的原因（比如忽视了语言特性或没有注意到细节）也有尚可原谅的原因（比如矛盾的或沟通不当的需求）。此外，bug 是软件发生变化的根源，因为当我们识别出 bug 时就会重构代码来修复它们，从而也会在这个过程中增加新的 bug。

早在1969年，迈尔·雷曼［Meir（Manny）Lehman］第一个指出软件在它的生命周期中会不断演变。后来他又指出软件开发的工作中一定存在多重的反馈循环，并且这些反馈循环会影响演变过程。它们是人们提出的大量（可能相互矛盾）的需求和设计决策。

程序员对需求、设计决策和实施细则不同程度的理解又会产生另外的反馈循环。换言之，bug 的来源不一定都是逻辑编程错误。观点的分歧也会导致 bug 的出现。

完美实现是一种妄想，它以为足够重视细节就能创建完美代码。如果是这样的话，我们都会坚决支持将编程技术结构化。但是我们不能，因为我们有坚定的理由。软件在其演变的任何阶段都存在 bug，极有可能会发生变化，并且文档记录也会出错。

尽管这一见解可能太简单了，但它能鼓励我们用不同的方式开发软件。它鼓励我们研发各种工具和技术来逐步重构软件的实现、需求和文档。

# 43. 敏捷的沟通系统



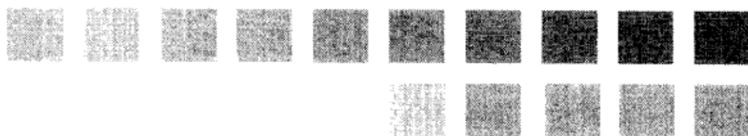
**布赖恩·萨姆–博登（Brian Sam-Bodden）**
美国亚利桑那州斯科茨代尔市

当反思失败的项目时，大部分责任都归于软件项目经理、团队成员和利益相关者之间的无效沟通。项目经理都知道要填平项目成员之间的沟通断层，并提供持续有效的沟通。对这一职责的重视程度有时会导致项目经理反应过度。他们会分不清哪些沟通是重要的、具体的，哪些沟通看似内容充实，但是却对项目进展有百害而无一利。

为了解决这个问题，许多软件开发人员正试图采用一种更灵活、更敏捷的方式。敏捷方法的关键之处在于及时的沟通循环，使敏捷团队能够有效应对未预见的变化，并且快速地重新评估和设置优先完成的项目功能。

敏捷项目经理们如何让沟通做到简明扼要？他们提倡每日"15 分钟站立"会议。这种会议要求开发人员讲述自上次会议后他们完成的任务、"今天"计划完成的任务以及在达成目标过程中他们预见的任何障碍。"站立会议"有一定风险，因为它完全依赖于每个开发人员自我评估的准确度。有解决办法吗？为了使站立会议更有效，要结合使用一个能显示测试结果的任务管理工具。工具对于项目基本代码的状况不会说谎，并且测试结果对开发人员的自我评估来说也是一个有价值的补充。提交已经通过一系列测试的某个功能的报告数据，也为这个功能的状态提供了更准确的描述。

例如，利用一种持续集成工具来描绘出进程的客观图像。这可以让"站立会议"的沟通只包含必要成分：障碍报告（希望这已经被任务管理工具捕捉到）和因边缘案例、集成困难以及 bug（缺陷）而造成的未能预见的发展情况。通过借助于一个全球共享的访问工具反映这些"新发现"，开发人员可以获得更精确的反馈。通常，在早期就可发现功能和任务之间存在着某些看不见的联系。



一个典型的误解就是同步沟通[1]总是比异步沟通[2]有效。添加开发工具和简短的异步沟通环路可以作为面对面沟通的有效辅助。

维基系统依据的是更广泛的反馈，轻易地就能让大家认识到项目开发进程的现实情况。这种系统还能够及时地提供信息，并为所有利益相关者提供更高层次的沟通渠道，因为这些利益相关者可能对阻碍某项功能开发进度的底层技术细节不感兴趣。相比之下，软件开发人员对项目整体的认识可能会被每天技术工作的细节淹没而逐渐变得模糊不清。维基系统可以让所有参与者保持清楚的共识。

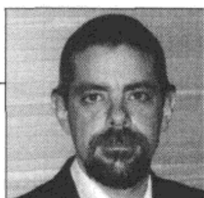
只要让信息沟通紧凑简短且不谈废话，软件项目经理就能够帮助避免沟通失败，而这通常又是项目失败的原因。项目经理的职责和挑战就是简化每一个项目层次的反馈环路，使之更高效。



① 同步沟通：不管是亲临现场，还是通过数字虚拟方式，所有参与者都同时参加。

② 异步沟通：参加者能获得信息，但不必亲临现场和进行实时联系。例如通过电子邮件、讨论板和共享文件夹。

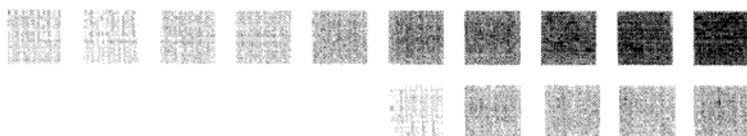
# 44. 不要崇拜方法论



**法比奥·特谢拉·德梅洛（Fabio Teixeira de Melo）**
PMP
墨西哥韦拉克鲁斯州夸察夸尔科斯市

许多项目经理过于推崇某种方法论，这会妨碍他们管理项目，使完成的项目不能成为实用且值得称赞的工作成果。如果你在最近的工作中使用过一种特定的方式，或者是在学校学的，或者是取得了资格认证的，那么你可能会想要严格地按照教科书所描述的那样去建立书里提到的所有进程和文档。这是一个危险的陷阱，在下述几个方面可能会引发各种问题。

- **所需付出的努力程度。**要想全面地完成参考资料中提到的所有流程，可能需要每个团队成员做大量的管理工作。你确定你估算的时间和预算里都考虑这些内容了吗？你当然不希望本来很顺利的流程到头来却因为要花费过多时间准备而导致项目失败。
- **执行公司的文化。**你的团队对那些专业流程的熟悉程度如何？你是不是得培训团队成员？培训有预算吗？他们感兴趣吗？你必须面对的职能经理和公司其他部门对此是怎么想的？你的流程与公司的建立正式或非正式的流程和习惯有冲突吗？这些冲突可能是该项目面临的风险。
- **项目重点。**项目经理所关注的一定是项目的成功完成，而软件项目的成功完成主要与交付相关。你所掌握的所有项目管理知识只是一种手段，不是最终结果。此外，你作为项目经理所重视的事情，你的团队自然也会重视。如果你关注的重点是充分建立和遵守所有的项目管理流程，那么你的团队也会重点关注这些。这样的话，谁来创建软件呢？



- **虚拟团队或地理位置分散的团队。**若你的所有队员并不是在同一个地方工作，那么制定并强制他们执行某些流程可能会很困难。由于他们使用了不同的硬件和软件技术，可能致使他们难以遵从你的要求，甚至是不可能遵从。当你的大项目需要一些距离很远的团队来为你提供交付产品或服务时，必要时请降低你的期望值。

最后，同你自己强劲的判断力相比，任何一种让你感觉需要被迫去遵循的项目管理书籍或方法都不重要。你应该先进行产品分析、合同分析、初次风险分析以及与主要利益相关者（客户和赞助商）的面谈，然后再选择项目管理战略。

尝试按自己的方式来加以评述，比如可以这样说："我计划用这种方式来管理这个项目，因为……"当你作决策的根本依据发生改变时，这将有助你调整策略。根据项目的需要，你能够确定哪些流程是最重要的，应该认真执行，哪些流程无关紧要。最后，好的项目管理计划就是高效、简单的计划。



# 45. 电子表格解决不了人的问题



**阿努潘·昆杜（Anupam Kundu）**
美国纽约州纽约市

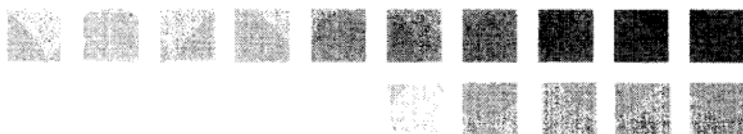
你是否得到过一张包含任务清单的电子表格？许多有经验的管理人员都希望电子表格成为一颗“银弹”[①]，企图用它来管理和监测各个项目。

汤姆在一家大公司的网络部门任 IT 开发架构师。他至少为该部门中四个不同的小组服务。他不善于为交付给不同小组的产品排出优先次序，因此，每周他都会让某些人恼怒或失望。由于有太多承诺要履行，而手头资源又太少，他总是成为各个小组间矛盾的焦点。

汤姆和他的团队成员都可以说是天才的 IT 架构师，但是，因为没有时间也不懂如何管理相关人员的期望，所以他们给网络部门中的每个人都带来了麻烦。有解决方法吗？找一位训练有素的软件项目经理来排定优先顺序，为汤姆和他的团队列出每周、每月或每季度要交付产品的清单。

项目经理与相关人员之间的讨论比较轻松，从而可以确定交付产品的优先顺序。然后，所有这些内部客户都来评估这个优先顺序。这种方法不仅能够帮助利益相关者设定正确的期望值，而且汤姆和他的团队也能获得一个经常刷新的任务表。这让他们可以专注于一周内最重要的任务。

① 银弹（Silver Bullet），在民间传说中，银色子弹是能够杀死狼人、吸血鬼、巫婆或者其他各种怪物的唯一一种弹药。这里，这个比喻意味着一种能够神奇地解决所有重大组织问题的新软件或技术。具有良好效果的任何解决方案都可以被看做银弹。



使计划行之有效的秘诀是不要为每个项目只提供一份交付任务的电子表格。相反，项目经理应该依据汤姆能够支配的资源（时间、资金和人员）总量，让所有利益相关者参与选择哪些功能是最重要的（或最有价值、最能获利的），让他们自己排出优先次序，从而为他们确定恰当的期望值。然后，依据网络部门其他小组的需求，每个小组都会明白自己在那个星期可以从汤姆那里得到多少产品。在为汤姆的团队制定工作计划的过程中，尤其是当必须为多个项目排定优先次序时，沟通是最有效的方式。

也许，汤姆和他的团队在上一个项目中曾与某个小组有过不愉快的经历。通过这种方法，即使“黑名单”上的团队也照样可以得到想要的工作成果，直到今后他们尽释前嫌。

总而言之，软件项目管理管的是人和人参与的过程。团队内部以及竞争性团体之间的人际冲突是很常见的。在想法、目标、价值观、信仰和需求方面的多样性应该是团队的主要优势，而不是弱点。然而，这些因素不可避免地会导致个人冲突以及团队在工作流程优先次序上的冲突。

大多数冲突会影响生产力和工作效率，圆满解决这些问题能切实地增进人际关系，促成创造性的改变，改善工作成果。所有解决冲突的策略都依赖于积极的沟通、主动的倾听、富有同情心的理解以及一些有效的谈判或仲裁。一名能力强的软件项目经理是必要的，因为你不能用电子表格解决人的问题。

# 46. 一件交付任务需由一人负总责



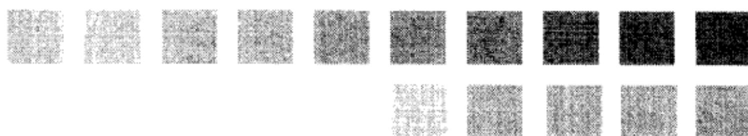
艾伦·格林布拉特（Alan Greenblatt）
美国马萨诸塞州萨德伯里市

每一件交付任务都应有一个对应的责任人负责它的完成。在一个项目里，应该清楚地知道负责每件任务交付的责任人。每件任务的实际开发过程可能会涉及很多人，但是，为了确保它按时完工，为了理解与之相关的所有技术细节，应该有一个人为其负总责。

通常，尤其是在政治性很强的环境下，责任都是分担的，特别是在项目刚开始、情况模糊不明时。人们喜欢对一个显然会成功的项目负责，没有人想负责那些肯定会失败的项目。在项目刚开始时，责任一般是分担的，因为在这时不可能完全清楚地了解需交付的任务和与之相关的风险。没有人真的愿意推动一个模糊不清的任务并为它承担责任。

有时，一件交付任务很有吸引力，会有一群人想负责它。但是，为了不破坏这种积极的氛围，管理层没有将特定的责任固定分配到一个人身上，因为担心这样做会使有些人感到沮丧。可是这样一来，日后的麻烦一定会不小。

首先，假如交付任务出现问题，一个负总责的人往往会提早通知团队，因为她知道她会被追究责任。当时间紧迫时，如果某项任务没有指定给某一个人负责，人们往往会指望其他人来处理出现的问题，结果就是三个和尚没水吃。作为软件项目经理，最终将由你负责处理危机。



其次，在项目进展的过程中，当无论遇到任何问题都能确切地知道该找谁咨询时，所有团队成员，特别是新加入的成员，将会效率更高。假如你有一个问题，你一定希望自己找到了处理这个问题的专家。假如这个专家不知道答案，他将会帮你找到答案。对于刚接触、还不太了解的问题，你就不必将时间花费在找答案的漫长旅途中了。

最后，有时项目没有达到预期结果（这种情况常常发生）。如果人们不能对他们的行为（或不作为）负责，你就永远也不能解决出现的问题，而且整个团队的活力也会受挫。当一群人浪费时间来讨论谁应该对这件“群体”任务负责时，那对整个团队的执行力而言，实在是最具破坏性的。

你不应该是不断归咎责任方，而应该是正确分配责任。并且当一件任务提前交付并且低于预算时（或者至少准时交付并且在预算内时），更应该知道表扬谁。

# 47. 知识完善的谬论



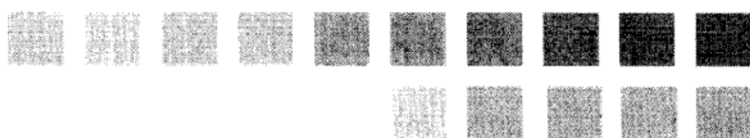
**戴维·伍德（David Wood）**
美国弗吉尼亚州弗雷德里克斯堡市

我们在内心深处都明白自己不可能是百事通。每一天，我们都希望能多学习一点专业知识、多了解一些我们的社会和我们自己。但是，我们的确不可能面面俱到。假如停止学习，我们很快就会落后于别人，特别是在软件行业里。一个人可以在一个行业里先做学徒，然后用余下的一生在这个行业里实践——这个观点就像渡渡鸟一样已经过时了。还记得渡渡鸟吗？不记得？这就是问题所在。

科学、技术及其背后的思想在这个年代变得太快，任何一个实践者都不可能在任一方面及时地掌握他所需要掌握的所有知识。我们必须不断地学习，同样地，我们必须将自己调整到一种无知的状态，在每个项目中都要花费一定的时间和精力来钻研我们需要的知识。既然如此，我们为何还要一再自诩我们必须甚至完全有能力在研发阶段就对这个项目了如指掌？

在过去，软件工程领域一直都充斥着对项目进行控制的企图，人们通过具有严密约束性的研发和维护活动，来防止出现有问题的、失败的软件。大多数这类方法论，如古典“瀑布”方法论，就是假设充足的时间和先前的勤奋可以使人完全理解一个软件项目。许多人要求在写代码之前，“需求”必须是板上钉钉，这简直是无稽之谈！

不要奢望在研发阶段能够无所不知，因此我们或许又认为在不久之后就会知道这一切。一些有关软件开发的方法论就是这样假设的，如螺旋或敏捷方法论。迭代开发被看成是让交付的项目满足“最终”需求的关键。不幸的是，对于那些方法论的支持者来说，一个软件项目的交付仅仅是开发过程中的逗号，而不是句号。



就算那些需求在先期详细的设计阶段得到“大家一致认同”，它们在开发过程中也将发生改变。预先获知一切是不可能的。多种多样的需求经常会相互冲突，甚至当这些需求出自同一来源时，也会出现矛盾。同样的需求对于不同的人来说甚至可能意味着不同的东西。不同的解释可能归因于认识、目标或语言的不同。为了创建一个成功的软件项目，我们必须接受甚至吸纳这些不同思想。我们无法无所不知，而且我们永远也做不到“无所不知”。

*知识完善的谬论*是一种错觉，这种错觉使人们认为在软件项目里可以获得完整的、没有冲突的需求。然而现实却是，在软件项目生命周期内的任何阶段，需求永远不会被完全了解，不管是在分析、开发还是维护阶段，甚至（或特别是）当这系统成为“历史遗产”时。

如果在项目生命周期的维护阶段继续使用迭代和重构等敏捷方法，就可以解决上述一些问题。不过接下来要更全面地了解软件演进的方式。直到我们有了那些概念工具，每天都使用它们，并且承认我们或多或少的无知，我们才不会继续成为知识完善谬论的受害者。

# 48. 培养团队跑马拉松，而不是冲刺



纳雷什·贾因（Naresh Jain）
印度孟买市马来德

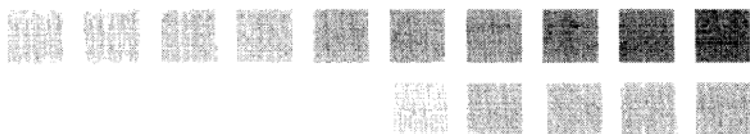
假如你在很短的时间内全速奔跑——田径运动中被称为“冲刺”——你将会使自己精疲力竭。为了跑马拉松，团队必须纪律严明，每天训练，保持一个可以持续的速度。在做软件项目时，我们也不想一次就弄得自己精疲力竭。我们需要保持一个稳定的工作进度。可持续发展的团队就像是在跑马拉松而不仅仅是在冲刺。

创建有用的产品并不是软件开发的终极目标。团队成员需要学习如何互相帮助、帮助他人认识自己真正的潜力、建立一个允许所有人突破自身极限的环境。

大多数团队对于这方面的认识还是一片空白。这就需要有人来积极填补这个空白。在大多数情况下，软件项目经理就是促成团队发展的最佳人选。我觉得项目经理的工作目标就是建立可持续发展的团队，他主要就是通过这种途径为项目带来附加价值。

假如项目经理重点关注团队建设和个人能力提升，整个团队自然就能不超时、不超支地完成交付任务。如果项目经理需要同时指导多个项目，这样做也能确保各个团队进行自我管理，而不需要保姆式的精心照顾。

通常，项目经理会陷入日常的灭火任务。因此，他们不可能真正有时间从战略高度来建立团队。如果对团队发展有长期的规划，项目经理就可以跳出微观管理的模式，不仅是在当前的项目中跳出来，而且会在未来的所有工作中做到高瞻远瞩。



我们需要从根本上改变软件项目管理的关注重点，以使项目经理扮演更具战略性的角色。将战术性的东西留给编程的团队，这将确保团队在项目中具有主人翁精神，而项目经理则成为真正的协助者或催化剂，确保这个项目整体朝着正确的方向发展。

乔治·巴顿（George S. Patton）将军说过："计划赶不上变化。"这意味着项目经理的大部分工作应该是培养团队处理不可预料的变化的能力，而不是参与日常编码和架构决策。如果项目经理认为自己能够欺骗信息技术团队，使他们相信自己在毫无编程背景的情况下就能掌握错综复杂的软件开发，那这样的项目经理就太愚蠢了。团队成员立刻就能知道这个项目经理根本不知道自己在说什么。

软件项目经理就像操作系统的内核[①]。内核本身不会完成最终用户的任务，但是它能确保用户的任务是由顶端的应用程序准确完成的。同样地，如果项目经理可以成为真正的推动者和教练，确保团队成员实现最佳合作，那么他一定可以建立一个能自我管理的团队，这个团队不仅能跑马拉松，还能交付高品质的软件。

① 系统内核：操作系统的核心部分。它负责硬件和软件之间的通信，将应用软件——比如互联网浏览器、文字处理软件、电子表格和电子邮件——连到电脑的硬件上，这样，所有的应用程序都可以使用内存、处理器和输入／输出设备。

# 49. 三位一体的项目管理



**保罗·瓦戈纳（Paul Waggoner）**
MBA，PMP，MCSE，CHP，CHSS
美国爱荷华州沃基市

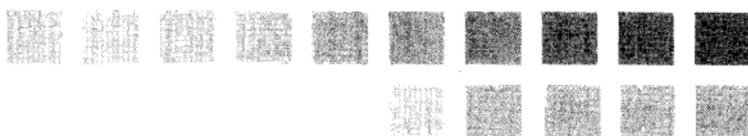
在新项目开始时，软件项目经理通常会详细阐述每位团队成员的角色，说清楚为什么每位成员的技能都至关重要，提出每个人应该准备完成的一般职责，并将这些内容都记录在案。然而，这些文件中都很少会去解释项目经理在项目生命周期内的角色。

项目经理面临的挑战是如何用 30 分钟概述项目管理的中心内容，还不能用那些具体的方法论细节吓倒整个团队，当和新团队合作时，这个挑战就显得格外严峻。

身处一个繁忙的公司，团队成员优先考虑的都是完成自己的主要工作任务，此时，作为项目经理，你面临的挑战是尽可能简洁地传达主要内容，而这些团队成员要从你组织的这次会议中了解的主要内容正是“三位一体”，也就是众所周知的三重约束。

为了介绍核心要点，你要准备幻灯片或者其他视频资料来描绘这三重约束。这是一个等边三角形，三个顶点分别是时间、成本和范围。位于这个等边三角形中心的正是项目质量。用这样一个几何图形来代表要完成的项目工作正是为了说明，增加三条边中的任意一边的长度都会迫使其他两条边（或至少其中的一条边）发生改变。而且，这种改变也会影响到项目的质量。

一旦指明了这三个约束条件之间的固定关系，就可以解释清楚为什么对范围的界定会是关键的第一步，而且也是一个主要限制条件。当然，依据公司项目管理的成熟度、具体管理的项目类型、项目管理能力的成熟度以及过去与客户打交道的经历，你可能会有自己的重点内容，不过，一定要包括以下这些方面。



- 每个团体成员的积极参与对整个项目的重要性，包括对制定项目计划的帮助。
- 项目风险——会遇到什么样的风险，怎样识别这些风险，为了避免、降低或应对这些风险，应该怎样做计划并监督计划的实施。
- 需要细化的任务，阐述团队中每个成员要完成的具体工作。
- 任务的分配、计划完成的日期、整个团队中各项任务之间的依赖关系，以及项目经理随后为了确保团队按时完成任务而需履行的职责。
- 可能出现的延期、有碍任务按时完成的因素、项目经理排除这些障碍的职责。
- 沟通计划、团体成员的沟通职责、项目经理协调计划细节的职责。
- 项目状态会议的职责和时间安排。
- 项目经理随项目计划展开后将要实施的工作大纲。

除非你的公司是一家成熟的、以项目为导向的企业，否则在所有项目刚开始时，一定要对基本的项目管理内容进行概括，它可以帮助团队成员充分理解他们的职责以及具体的支持细节。同时不要忘了介绍你作为项目经理的职责。

# 50. 路线图：最近我们为你做了什么



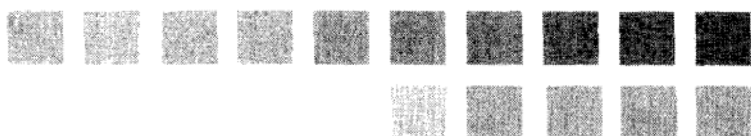
**凯西·麦克杜格尔（Kathy MacDougall）**
美国科罗拉多州伊利市

对于任何项目而言，良好的内部和外部沟通都是成功的关键因素。在所有的项目中，重要的沟通工具就是正式的项目路线图。项目计划可以帮助你手下的项目团队定位任务阶段的变化。相比之下，项目路线图能够使更广泛的利益相关团体充分理解可能在更高层面上发生的变化。项目路线图这种工具有助于传达计划之中的变化、特定变化的时间表以及这些变化将会对工作造成的影响。

那么怎样制定项目路线图呢？首先，列出项目主要利益相关者提供的信息。对他们来说，什么功能重要？每项功能的优先级别是什么？有没有什么因素决定某些功能必须在某个特定日期前完成？要倾听客户的需求，并以此为基础起草一份路线图。

接下来，草拟一份路线图，按照现实的时间表（通常是按季度）列出高层功能。在路线图上标示出每项功能的商业价值（例如，将订货时间缩短两分钟，将订货费用减少 10 美元）。如果说不出商业价值，那么你就应该考虑该项目是否需要实现这个功能。简而言之，如果一项功能没有切实的商业价值，它就不应该出现在你的路线图上，对于这样的功能，我们一定得采用成本 / 效益分析来做进一步的审查。

一旦创建出一份合适的草图，就需要从项目主管者和利益相关者那儿获得反馈信息。开辟一个论坛供利益相关者讨论，他们可以要求解惑，说出对优先顺序的顾虑，或者提醒团队关注没有在路线图上出现的功能。这些坦率的讨论能够加强对整体项目的理解，并有助于确保这份路线图符合利益相关者优先考虑的事项。根据收到的反馈信息调整这份草图。从理论上讲，在完成这步之后，你的路线图将获得所有主要利益相关者的支持。



最后，大张旗鼓地把路线图公布出来——把它放到项目网站上，将它介绍给次要利益相关者，并把它作为项目的主要交流工具加以使用。每个季度检查一次路线图，以确保你还在正轨上前进。告诉利益相关者哪些功能已经完成了，哪些将在下个季度完成。假如因为某些工作的延误而不得不改变路线图，那么就回到草图阶段再重复制定一次路线图。与项目所有相关人员就最新修订的计划进行沟通。

这种建立项目路线图的方法使利益相关者有了话语权，而且让他们知道该期待什么。最后，同样重要的是，路线图让团队能够定期对外公布每个季度已成功提交的功能。



# 51. 项目范围说明的重要性



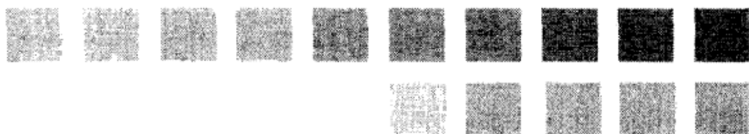
**金·海德曼（Kim Heldman）**
PMP
美国科罗拉多州雷克伍德市

如果说项目计划是一套项目管理好方法的心跳，那么范围说明就是它的呼吸。范围说明详细阐述了项目的愿景，它描述了目标和交付成果，并诠释了怎样才算项目最终的成功。

遗憾的是，许多利益相关者对编写范围说明没什么兴趣。更令人深感遗憾的是，虽然大多数项目经理的确是花费了时间来创建一个面面俱到的范围说明，但当他们拿回签了名的说明书后，却会迫不及待地将其束之高阁，上面的墨迹甚至都未干，并且从此都不会再看一眼。在项目的实施过程中，一定要不断回过头来依据范围说明进行检查，这样做可以确保你所交付的东西正是客户所期望的。

我很喜欢打比方，我经常和客户提到的一个比喻就是房屋改造的故事。假设你已雇用了一个承包商来装修你家的地下室。问题在于，是你直接告诉承包商该怎么做，还是让他自己决定如何布局呢？当然，承包商可能应该在建筑布局上有一些自己的想法，如房间应该多大，管道在哪里。但是，如果你想要两间卧室、一个带淋浴的卫生间和一间游戏室，而承包商所建的却是一间卧室、一个带浴缸的卫生间和一个有完整吧台的起居室，这可怎么办？根本与你想要的背道而驰。

这充分说明了范围说明的重要性。它是项目的蓝图，定义了项目的最终产品或服务的所有特征。如果没有它，当利益相关者期待有两间卧室时，你却可能只建造了一间卧室。范围说明可以帮助你满足利益相关者的期望。不止一次，在项目开始了一段时间之后，通常是几个关键的最终交付物都过了雏形阶段，这时突然有一个利益相关者说："我还以为我们正在建造两间卧室呢。"在这时就需要范围说明来救援了。你当然不能把它当作一件武器挥来挥去，但是它能够温柔地提醒利益相关者，不要忘了他们在项目开始时就同意的内容。



定期和利益相关者回顾一下范围说明也是一个好习惯。项目状态会议或指导委员会会议就很适合进行这种回顾。每隔一次（或依据项目的大小和范围来确定合适的间隔时间）就在会议上抽出一定时间来回顾一下范围说明上列出的最终交付物。

如果你正在定期召开项目状态会议，某种程度上你可能已经开展了这项工作。状态会议通常会汇报上个阶段已经完成的工作，以及下一个阶段将要完成的工作。你需要时不时花些时间来越过下一个工作阶段，提醒众人不要忘了之后要交付的所有关键任务。

定期审查项目范围说明可以提高项目成功的几率，并且使利益相关者的期望与该项目的目标保持一致。



# 52. 愿景与预期结果保持一致



**戴维·迪亚兹·卡斯特罗（David Diaz Castillo）**
MBA，PMP
巴拿马巴拿马城

软件开发项目是非常富有挑战性的，因为对需求和期望的定义通常都很模糊。软件项目经理的工作是确保做到以下几点。

- 明确界定项目的主要目的。
- 让每个人都明白为什么要开展这个项目。
- 清楚地认识到对三个 P（person、process 和 platform）的影响力。
- 要求和期望都应列入需求文档中。鉴别哪些需求在范围内以及哪些超出范围，然后就此内容与团队成员进行沟通。

软件项目经理需要让团队成员在愿景和预期结果上保持一致，此外还要掌握下面三点内容。

(1) **商业视角**。这个项目为什么是解决方案？（这个项目要解决的是什么问题，或要提供什么样的机遇，或如何给组织增加价值？）

(2) SMART **视角**。这个软件要做什么？［要明确（Specific）、可衡量（Measurable）、获全体同意（Agreed upon）、具现实性（Realistic），并且可以在时间期限（Time constraint）内完成。］

(3) **主观视角**。最终用户认为该系统应该做什么？（在初始阶段就获取最终用户对该软件的期望和理解。）



**第一点**。编程工作一开始，设计团队和软件项目经理就专注于项目的设计功能和技术部分，而不理会公司投资研发这个项目的主要原因。由于没有时刻提醒团队成员关注他们真正需要解决的商业问题，因此在研发过程中团队会对已经做出的决策产生误会、理解不当甚至理解错误。项目应该带给公司的利益并非被大家放在第一位。为了避免出现这种情况，项目经理需要让大家都清楚项目的目的、前提、约束和风险。

**第二点**。项目的技术和设计功能必须非常明确，足以让所有团队成员领会，包括项目的资助人。项目成果必须符合最终使用该系统的商业领域的战略目标。

**第三点**。项目经理应该确认最终用户的期望。用户认为这个新的应用软件应该如何为他们的日常工作提供帮助？项目经理必须清楚认识到这些好处和期望，并将这些内容传达给开发团队，使团队成员都能真正理解并接受。在弄清楚这些问题之后，项目经理才可以进一步将这些好处准确地传递给最终用户，帮助他们形成一个有关最终产品的现实目标。

详细掌握了项目的目标和价值，实时决策就变得更加容易。此外，由于项目经理真正知道用户对项目的期望和研发该系统的真正用处，他才能够更有效地评估变更控制提交单。这样可以防止项目在执行阶段偏离正轨。

作为项目经理，我们应该严于律己，自己要真正理解项目的技术需求和要提供的商业价值。只有了解了这些，我们才能做好准备去创造更好的软件产品，并且以更加专业的方式管理项目生命周期中的不确定因素。

# 53. 艾丽丝不是美国人了



**芭比·戴维斯（Barbee Davis）**
文学硕士，PHR，PMP
美国内布拉斯加州奥马哈市

目前，软件开发人员最爱讨论什么是最好的编程语言、系统架构、操作平台或项目方法。似乎没有一个人注意到我们团队中的某位成员——艾丽丝——并不是美国人。现在的艾丽丝是哪里人？这种改变会如何影响我们的软件开发计划呢？

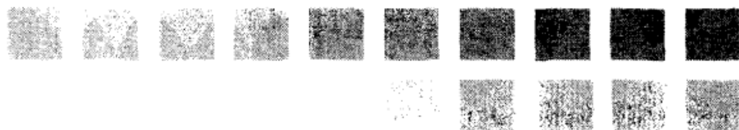
她可能是印度人，在印度英语可能是通过他们自己的一套印度拼音系统来进行语音训练的。因此，要为沟通预留出额外的时间或采用书面交流方式，这样才能不惧语言障碍，让艾丽丝能够顺利完成她开发的软件部分。

艾丽丝也可能是非洲人。由于那里缺乏技术人才，对雇主来说人员可能比项目更重要。技术条件也是有限的，所以不要以为 24 小时都能通过电子邮件、电话、因特网和她联系上。

又或许，工作出色的艾丽丝住在一个发展中国家。如果她在电话会议上反应有些迟钝，那么你要想到，在你说话和她听到你的话之间可能存在 30 秒的卫星通信延迟。在她说话的时候你也会遇到同样的延迟。

如果你有团队成员在日本，那么，你应该了解不同的决策方式。对长辈和有经验的人要给予更多的尊重。年轻的艾丽丝如果大声说话可能就会被视作不得体或有冒犯的意味。不论会议做何决策，日本的团队成员可能只有在全体通过之后才会将这些信息记录下来。

如果你有许多艾丽丝分布在不同的远程地点，你就需要仔细研究许多细小问题，这样你的团队才能顺利运作。



- 在艾丽丝工作的地方工会都有哪些规定？她的作息时间有什么不同？她可以加班或者在周末工作吗？她可以早来吗？她工作时可以没有午餐时间或工间休息吗？
- 在艾丽丝的国家公共假日都在什么时间？当你安排重要的集体会议时，你一定要留意会议时间是否适合每个人。比如说，你不能要求一个美国员工在圣诞节早上来上班。
- 在艾丽丝的国家会计惯例是怎样的？你要求的工作月报时间与其他地方的工资支付时间相一致吗？
- 会不会有意想不到的数据输出控制？有些欧洲国家对数据传输实施严格的控制。你也许能够将信息发送给艾丽丝，但是她能将数据反馈给你吗？
- 客户服务的做法可能因国家而异。如果艾丽丝的工作是要直接面对客户，那一定要确保她了解整个团队的服务标准以及对服务质量的要求。
- 如果你因无法使用语音邮件而备感沮丧，那么应体谅一下，要知道，一些发展中国家会认为，正是像语音邮件这样的自动化机器抢占了某人急需的工作岗位。当你在技术方面发现文化差异时要懂得尊重他人。
- 你总是希望艾丽丝能在半夜参加你每个星期二上午 9 时（你的时区时间）召开的会议吗？为了表示对所有团队成员的尊重，你可以寻找最不令人反感的会议时间，或让各地区轮换不方便的会议时间。

我们很幸运能拥有一个虚拟团队的伟大思想和精辟观点。请一定要恭敬地使用这笔得之不易的财富。

# 54. 避免合同纠纷



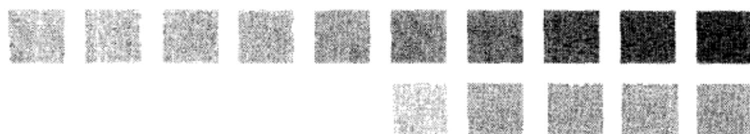

**乔治·格拉伯特（Jorge Gelabert）**
PMP
美国康涅狄格州柏林市


获得 PMP 资格认证的项目经理对各种类型的合同都很熟悉。在选择合同类型时，他们知道不仅仅是依据购买的产品和服务，而且还要依据他们和卖方愿意承担的风险等级。有一点他们可能永远都没有意识到：即使是最好的合同也无法担保不出现纠纷。

为了避免可能出现的纠纷，人们很容易会想到对需求要给予明确的定义。如果在合同中能够明确期望，那么合同双方就会对最终交付物达成共识。然而，在项目管理的实际操作中，并不是总能对需求给出明确定义。一些销售人员（你的销售团队）为了做成生意，也许会提出一个低于预期的提案，因为他们认为客户一定会出现需求的变化，而他们届时就可以利用这些变更订单来挽回最初报价中的利益损失。即使项目出价很好，并且合同双方对项目范围也有严格、明确的一致规定，但仍有可能会出现双方都必须应对的变化。这些情况，再加上其他可能出现的种种特殊情况，都可能成为产生纠纷的领域。

那么项目经理能做什么呢？在管理项目时，心中应该想：客户是搭档而不是对手。假如卖方（你）和买方（你的客户）都希望项目成功，那么产生的争议会很容易得到友善的解决。假如双方都过于关注自己的利益，那么很小的争议都会发展成大的争端而使项目脱离正确的轨道。

当发生争议时，要通过双赢的方式来解决。即使合同对你有利，也要进行协商。虽然你可能会觉得你能够在法律上证明你是对的，但是这样做会伤害双方的关系，就算官司结束了，双方关系的恶化以及显而易见的项目僵局都将有碍项目的最终成功。只要还有一线希望，就不要使用法律手段，那是万不得已而为之的举措。



避免产生纠纷的最好方式是保证合同的公平公正。比方说，如果合同约定了违约金，那就一定还要约定奖金。让双方都感觉到他们各自的得失是平等的。即使是一份已经确定价格的合同，重新协商也比让买方（你的客户）放弃这个项目要好得多，一旦客户认为他们的项目得不偿失，就会做出放弃的决定。

此外，对好得令人难以置信的出价一定要保持谨慎。假如你的销售团队给出的报价并不具有竞争性，那就意味着你的买家并不完全清楚完成项目必须要做的工作。一旦他们意识到自己的付出大于所得，他们很有可能要求重新协商或者干脆放弃这个项目。



# 55. 评估什么，就得到什么



**纳雷什·贾因（Naresh Jain）**
印度孟买市马来德

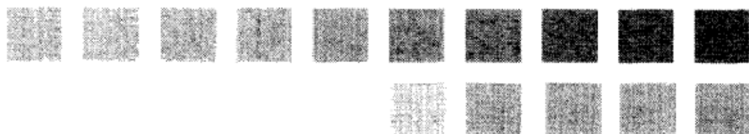
众所周知，如果你以错误的东西作为评估指标，你就是在鼓励错误的行为。经理用不恰当的参数追踪和评估团队成员的工作，会使团队成员们每天都备受折磨。

比如说，用工作时间作为评估指标就会鼓励一些团队成员工作更长的时间。研究显示，延长工作时间未必会产生更好的工作结果。在大多数情况下，延长工作时间实际上会造成工作质量更差。

同样，关注且评估团队的工作速度（在一定时间范围内团队实现的功能数量）会鼓励大家以更短的时间来完成更多的工作，但是这并不一定保证大家就会选择最至关重要的工作去做。因此，这个方法并不利于又快又好地完成软件开发。

重视测试者检测出的 bug 数量，会鼓励测试者提交更多的 bug。但是，那些严重影响工作的问题，测试者却并不一定会发现。如果将被检测出的 bug 数量作为评估开发者的标准，那么测试者将成为开发人员的敌人。这就导致了团队关系不必要的紧张。

以我的经验来看，做得越多、做得越快，并不意味着这套软件就是成功的。软件研发得快有利于快速得到反馈，但是创建真正的产品所需要的不仅仅是研发速度。



看到运转不正常的团队时，我常常发现产生这种不和谐是因为团队使用不恰当的指标进行评估。久而久之，团队就接受并适应了这些选择不当的评估指标。由于缺乏对项目目标或愿景的理解，团队成员就会自己定义一套成功标准，并且用自己个别的、孤立的、不正常的因素作为评估指标。错误的评估方法有百害而无一利。

好的项目经理要确保团队里的每一个人都能真正理解成功意味着什么。他们要帮助团队成员建立一个共同的愿景和共识。他们通过建立双赢局面鼓励团队成员间的合作，以使每个成员都有共同的关注点，并朝着共同的目标前进。他们帮助团队成员确认什么是真正需要被评估的。成功项目的秘诀是利用衡量指标作为达到目的的手段，而不是使衡量指标本身成为交付的任务。

我发现，如果打算一次评估 10 种不同的东西，就会使团队成员变得迷惑而不知所措。一次只评估两三个指标，则会很有效率。为了确定这两三个评估指标，应该看看目前危害团队的问题或在以后将会产生影响的问题，然后由团队成员一致决定。

一旦解决了某个问题或者缓解了某项风险，团队应该在评估体系中删除这项过时的指标并增加新的指标。一个不会定期改变评估指标的团队一定会遇到更大问题。

确保你所评估的指标是有意义的，而且还要知道，这项指标在项目进行中是会发生变化的。评估什么，就得到什么，因此，要确保你正在评估的指标没问题。

# 56. 他山之石，可以攻玉



**保罗·贾马尔沃（Paul Giammalvo）博士**
CDT，CCE，MScPM
印度尼西亚雅加达市

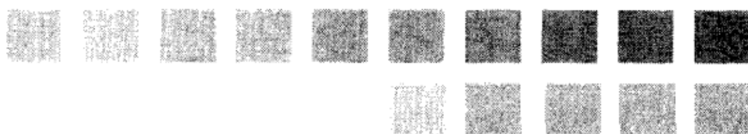
项目管理不过就是一套流程，对这些流程加以整合，就形成了方法论。而这些流程或者说方法论几乎是通用的，对应用领域没有限制。

下面有5组与项目管理有关的流程。

- **启动**。这组流程授权或承认一个项目的存在。
- **计划**。这组流程使我们能够确定需要做什么以及如何去做。
- **执行**。对计划阶段确定的流程加以执行，用以创造成果。
- **监督和控制**。在这组流程里我们要评估项目是否正在按照计划进行。
- **收尾**。在这组流程里，我们要确认是否按时完成项目，是否在预算经费内，是否完全达到要求，是否实现了预期效果。

IT领域的人特别不愿意观察别的领域中项目管理都是怎么做的。对于那些更先进或者更成熟的领域，IT领域的人似乎不会去借鉴或采纳他们的“最佳实践”来提高IT项目的成功率。

可以作为标杆的两类优秀领域分别是医疗和民航。为什么呢？因为医疗和民航在他们的交付系统中都体现了项目管理。在医疗行业里，交付的是每一次的手术或治疗过程，在民航业里，交付的是每一次从A点到B点的飞行。但是，从IT的角度来看，还有一点更重要，医疗行业和民航业的成功率是非常令人羡慕的。



那么医疗行业和民航业到底拥有了哪些独门秘诀？首先，医生和飞行员在做决定时，他们几乎有至高无上的权威。不过这也意味着，在这些岗位上的人要承担所有责任，既有经济上的，也有职务上的。特别是飞行员，他们一旦出错，就是拿自己的生命在冒险。

其次，医疗行业和民航业都不能接受“一般水平”的表现。PMI 的 *PMBOK® Guide* 指出，项目管理知识体系是指那些“公认为好的”技能、工具和方法。

最后，应用在医疗行业和民航业的项目管理不是独立的方法论。项目管理之所以在这些领域存在，并且很成功，主要是因为项目管理与资产管理（对项目资金的分配和使用负责任的部门）和运营管理（通过为公司处理日常工作来获得收入的部门）有着充分的不可分隔的联系。如果没有资产管理部门提供充足的组织资源，没有作为内部客户的运营管理部门提出符合实际的期望要求，IT 项目就不可能在任何一个公司里获得成功。

软件项目经理必须乐意看看 IT 领域之外的世界，学习其他领域项目管理的成功之处，特别是医疗行业和民航业，这两个领域的成功率比 IT 项目显然要高得多。

# 57. 要现在不要马上



**斯科特·戴维斯（Scott Davis）**
美国科罗拉多州布鲁姆菲尔德市

我很喜欢一句名言——“夸张比保守糟糕百万倍”。对于项目管理，就可以将这句话改成“现在比马上好百万倍，比过会儿更要好千万倍”。

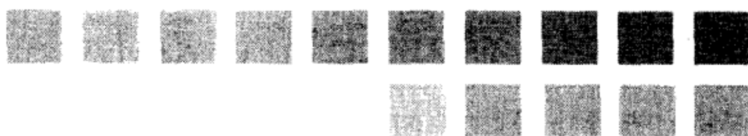
如果你从事的是软件开发业，那么你该熟悉“雾件”现象——即某个软件不断地被提及，但却从来没有实际交付。我们可以计划软件项目，我们可以讨论它将具备的特征，但是一个实实在在可以运行及交互操作的软件远比仅仅拥有一个充满大堆需求的文档要好百万倍。

这意味着赶快编写软件——就在现在！你建成的原型会立刻向你反馈有关可用性的信息。担心性能吗？没有软件在手，优化性能又从何谈起？

敏捷开发法非常重视快速迭代，因而更倾向于现在而非马上。通常一次迭代的周期不超过一两周。这种方法的主旨思想就是为了让软件尽可能快地写成并到达用户手中。如果用户看过之后表示喜欢，那么你立刻就成功了。更重要的是，如果用户不喜欢，那么你也很快宣告失败。

快速成功仅仅比迟来的成功好上一百倍，但快速失败可比迟来的失败要好上百万倍。它使你有时间反思、调整，尽早重写软件。在交付日期前一星期失败就像被掌声打断的精彩演说，令人刻骨铭心。但研发过程早期的失败会随时间流逝而慢慢被人淡忘，等到软件实际完成的时候，那次失败早就被人抛之脑后。

如果你使用的软件语言或框架不能在几秒钟或几分钟内实现新功能，那么你正在使用的工具就是一个问题。如果编译代码需要几个小时而不是几分钟或几秒种，那么你就做不到早构建、常构建。这些软件开发中的阻力点会使你处于明显的竞争劣势。许多你天天使用的顶级网站可以在短短 30 分钟内拿出新功能。



他们之所以成为这个行业的领跑者，也得益于他们迅速生产的软件的质量。他们不停地测试自己的代码。难道他们会等到软件开发末期才编写他们的单元和集成测试吗？不，他们会立即写。测试优先和测试驱动法受到顶级软件开发人员的青睐，这是因为，如果测试十分重要，那么就值得现在做。

这并不是说计划在软件项目中就不重要了。这只是说你要根据现代软件方法和要求来做计划。方法论可以追溯到很久以前，那时代码是手写出来的，再被精心转移到打孔卡上，然后用手将其放到一个方盒子里，交给系统管理员。在现在这样一个简单、自由、即时的软件时代，那些过去的方法论并不合时宜。我们正处于“就是现在”的时代，你的方法也应该做出相应的调整。



# 58. 速度就是生命，越快越好



**马特·“布姆”·丹尼尔（Matt “Boom” Daniel）**
美国宾夕法尼亚州库珀斯堡市

“速度就是生命，越快越好”这句话在喷气式战斗机的团体里是一个常见的口号。身临其境地想一想，我们是不是经常能听到“要快”，“一切事情都必须以最快的速度进行”，“现在就去，现在就走”？

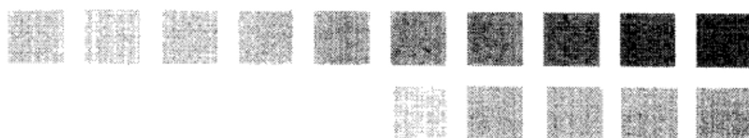
毫无疑问，在战斗机飞行员的日常飞行中，速度是一个基本要求。[在电影《Top Gun》中，小牛和鹅（两个战斗机飞行员的外号）就是这样说的，那么这句话一定是真的。]

但是，“速度就是生命，越快越好”这句话永远都对吗？

在经典的“一对一”对抗中，降低速度以减小转弯半径就是一种可行的策略。你转一个较小的圈，同时迫使对手飞一个更大的圈后出现在你的战斗机前，这样你就有一个更好的射击位置。你“要住在他的飞行圈里”。进行航空表演时，当这两架飞机都像棒球比赛里的快速球一样飞行时，能做到这一点才算是真正的控制。

科学研究证明，对于特定的行动、战术和表现形态而言，具有优势的是最佳速度，而非超常速度。我们的目标就是最佳速度，而不是最快速度。所以，一旦选定某种需求或战术，速度仅仅是一个关键的衡量标准，更重要的是，你选择怎样使用这种能量（速度）。

现在不谈战斗机驾驶员，来说说危机四伏的商界，第一家推出新技术的公司永远都是赢家吗？如果目标仅仅是开发一种有生命力、有价值的产品或服务，那么对这个问题的回答顶多也就是“可能”。



在公司的商业计划里，“第一个上市（速度）”也许根本就无关紧要。首先进入市场者因为过于关注能量（速度）而忽视能量管理（只有当某项业务功能的确需要速度时才运用高速）并最终破产或牺牲的例子，在技术领域里数不胜数。

- **通信卫星**。铱星电话因为一些更简单、更便宜的通讯系统的出现，而不再受到普通人的青睐。
- VCR（**盒式磁带录像机**）。Betamax 录像机开发早于家用录像机（VHS），是首先推向市场的高端产品。由于该公司拒绝对其产品、服务和衍生品进行专利权交换，致使这项技术过时。
- PDA（**个人数据助理**）。苹果电脑公司的苹果牛顿数据助理，尽管很早投入市场，但最终还是敌不过交互式的 Palm 电话。
- TV-to-Web（**电视网络机**）。WebTV 是早期的创新产品，它用电视作为显示器，而不是用电脑显示器。因此它永远不会流行。

问问自己，作为一个软件项目经理，如何既能保证发布速度又能确保长期存在的价值？使用什么工具或方法能确保你的新解决方案不会过时？

你有速度至上的观念吗？它究竟是优点还是缺点？在你的环境中，速度代表了什么？能量管理对你的项目团队意味着什么？

# 59. 激发团队士气

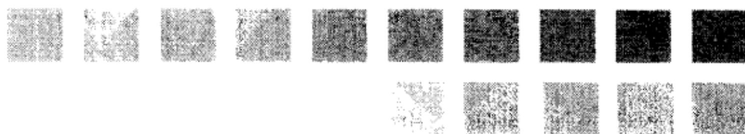


**戴维·博克（David Bock）**
美国弗吉尼亚州雷斯顿市

软件项目经理的一个主要职责就是，创造一个良好的工作环境，激发团队士气。下列这些技巧能帮助你开启激励之门。

- 对团队在项目发展方向上加以控制。你跟团队交流频繁吗？你经常征询他们的意见吗？有人会提议或者向你抱怨吗？有人会觉得你将因为他的抱怨而做出改变吗？
- 保护团队免受“官僚制”影响。每家公司都有需要一致遵循的规章，你的工作之一就是根据具体情况实施这些规章。“根据具体情况”意味着在适当的时候你要保护你的团队。

  当一个公司出台禁止隔间装饰的规章时，你会据理力争，从而使比尔能够继续在办公室展示他收集的魔方吗？即使你输了，团队的士气也会因此而受益。
- 想办法改善工作环境。我曾认识一名工程师，他的工作隔间就挨着窗户。但是，根据公司的组织结构，他无权使用一个有窗户的办公室。公司如何解决？这些隔间要重新规划，安装上挡板，这样那扇窗户就被挡在他的隔间之外。一位优秀的经理不必去跟“家具警察”争执，而应该第一个走过去移开这些隔墙从而露出窗户。
- 让团队成员感觉他们的确像一个团队。一个团队要设立每周最佳队员奖，而且这个奖项每周都要在团队会议上轮换。拉斯可能会说：“我将最佳队员奖传递给玛丽，因为她在星期四晚上工作到很晚。虽然我把文件给她晚了，但由于她的努力，我们仍然在星期五的早上完成了迭代。”下一周，玛丽将会指出另一名团队成员的贡献，并将这个奖传给他。



- 保持工作和生活间的协调关系。你可以偶尔要求人们加班，但是如果你打算从人们的生活中占用时间，你就需要日后再把这些时间还回去。你的员工不应该害怕在早上预约医生或下午去参加孩子的家长会，特别是当他们前不久刚刚为了按时交付而工作到很晚。
- 有因必有果。如果你仅仅是想重现过去曾激励过你或其他人的某些做法，你就会错失要点。你应该问自己“我能做些什么来改善团队的工作条件呢”，如果你真的能努力做出这些改变，那么团队士气通常就会得以提升。
- 确保团队成员能看到你的努力。你也是一名团队成员，因此整个团队需要知道你正为团队所做的工作。团队成员往往不会信任一个总是藏在门后的经理，也很容易跟随那些有目共睹在为团队争取利益的人。

你的公司一定会有提高员工士气的独特机会。你要有意识地寻找它们，并利用它们。如果这些做法成功了，别忘了与他人分享。



# 60. 项目要依靠团队合作



**莱利奥·瓦芮拉（Lelio Varella）**
PMP
巴西里约热内卢州蒂茹卡市

项目是多学科性的事业。它可以被看成是集体努力的结果，由各种不同的人来共同完成。只要根据团队成员各自独特的技术和能力来部署工作，成员之间也搭配得当，他们就能创作出重要成果，并实现项目利益相关者的预期目标。举个例子，如果是一个 IT 方案的研发项目，我们就要想到各项具体的活动和不同的团队。

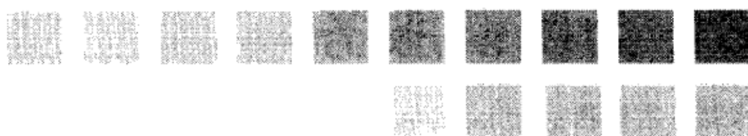
- **活动**。业务流程回顾、IT 解决方案和服务的定义、产品和服务开发以及新进程和服务的激活。
- **团队**。客户业务团队、IT 研发团队、外部服务提供商团队。

项目活动就是需要公司内不同团队人员参与实施的各项工作单元，最后也需要外部组织的参与，如供应商和服务提供商。

这些团队是由具有不同知识和能力的人组成的，实际上，就是由这些人管理、执行项目活动。每个团队的成员都会涉及项目内某组独特的活动。

为了让大家高效工作并创作出优良的项目成果，工作责任的分配应该直接与个人挂钩，而不是与这个人所属的部门或机构挂钩。这些责任分配应该被记录在项目计划书中。

根据个人的工作能力以及工作性质，每个人可以在一个项目中同时扮演不同的角色。一个人可以在此处是一位领导者，而在别处则是一位参与者。



每个团队对项目的参与和贡献是通过团队成员完成的集体活动来衡量的。既然个人的责任可以用一个责任矩阵来表示，那么，也应该为参与项目的每个团队的共同愿景和整体责任给出相应的责任矩阵。

每种项目都存在多样性，而软件研发项目中的多样性比其他任何领域都明显。团队合作取决于两个主要原则，即委派和责任，只有这两点能够共存并合作，才能取得成功。

委派应遵循项目工作分解结构所明确的方式。首先分解需要做的工作，然后进一步分解每项工作任务下的子项目。这是人类理解、执行、管理重大项目时唯一可行的方法。

既然要委派任务，你就需要对每项任务所需的技术和管理能力进行综合考虑。一旦委派别人去做，就不要干涉。作为软件项目经理，你要做的就是进行监管、提供支持、询问结果。这样做，你就可以激励团队、赢得尊重并培养团队成员的“反应能力”。

“反应能力”包括主动承担全部的责任。在进行委派以及接受分配时你要记住这点。个人的技术和能力要与需完成的任务相匹配。

一旦领导体系、委派及责任都安排到位，团队成员就要互相支持。这样才能保证获得更好的结果，并最终使项目取得成功。

# 61. 为团队服务



**卡伦·吉利森（Karen Gillison）**
美国弗吉尼亚州利斯堡市

在我尚未听说敏捷方法[1]之前，与我一起工作的项目经理是我当时所见过的最好的项目经理。现在回想起来，他那时使用的正是敏捷方法的雏型。他认为他的工作就是推动团队，他的日常职责就是识别和消除障碍，并提供团队资源。他所做的事情都是在提高团队速度[2]。

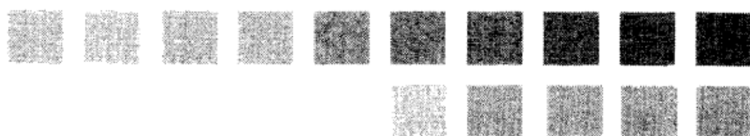
没有长达数小时、让你昏昏欲睡的会议（在轮到自己提供状态更新之前，还得努力保持清醒），这种会议让你不想去参加，只想回去写代码，以便下一次开会的时候可以汇报一些进展。相反，每个项目开始时，我们都会召开一个启动会议，从需求到测试，凡是与这些职能相关的人员，我们都会邀请来参加会议。整个团队聚在一起以获得对愿景和项目的共识。然后，每隔几天项目经理会来召开我们所说的“门口”状态会议。团队每个成员要对已经完成的工作、正在进行的工作以及重要事项做简短汇报。

项目经理以一种人人看得见的方式记录项目状态。他有一份标注所有工作任务的列表，表上列出了每个团队成员应该完成的工作。项目经理会定期更新这份文件，并用一张大纸打印出来张贴在他的门外。将信息张贴到所有人都能看见的地方，对于团队的沟通十分有利。此外，这样做还有一个好处，就是高层管理人员也可以看到这些更新的状态，而且，只要想看，随时就能看，都不必麻烦他人。

---

① 敏捷方法：是一套还在不断发展的方法，这套方法旨在推动软件项目管理流程，它提倡缩短计划阶段、提高对变化的适应性、团队工作、单元测试、个人当责以及客户的频繁参与。

② 速度：敏捷软件开发术语，表示团队或团队成员的进展速度，即一个程序员在指定期限内能够单独完成的工作量。



再谈谈“自负”。我最喜欢的这位项目经理很成熟，而且自律性也很强，因此不会表现出自负。即使身为老板，他也不会仅凭一时兴起就滥用权力、改变任务或转移工作方向。他的行为从来不会对团队的生产力造成损害，因为他的主要目的就是为了推动团队走向卓越。

通过克制自负的表现，他和团队取得了令人惊讶的成果，所有项目都在预算和时限范围内完成，令最终用户和高级管理层都相当满意。由于管理方式非常有效，所以在项目的最后阶段，根本不会看到有人通宵工作、大吼大叫或紧张无比。在不太成熟的企业中，这样的项目经理和顺利执行任务的团队可能不会得到认可，因为每个项目看上去都很容易。即使我们没有得到感谢，公司、最终客户以及每个人都通过实际行动对我们表示满意和认可。

今天，敏捷方法可以提供一些新式工具，使你成为更有效的项目经理。我建议你要熟悉这些方法，哪怕你还未有幸工作于一个已经采用了这种方法的组织中。现在就可以将这些工具融入传统的项目管理方法里。要知道，项目经理的一个关键作用就是提高团队的速度，要努力为团队的生产力创造一个没有阻碍因素的环境。

# 62. 大圆球谬论



**戴维·伍德（David Wood）**
美国弗吉尼亚州弗雷德里克斯堡市

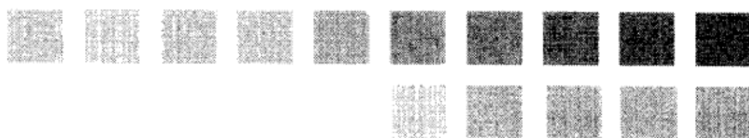
有这样一个球，它需要制作得非常圆、非常光滑。对这个球的唯一设计要求就是，无论在球面的哪一点上测量，它的直径都要完全一致。这个球被打磨、打磨、再打磨，直至非常完美。一旦找不到缺陷，对这个球的所有工作就会停止。它不再发生改变。它就是完美。

这听上去是不是像你做过的软件项目？恕我不这么认为。软件可不是这样实现的。

软件在它的生命周期中不断变化。设计决定往往是基于最初需求形成的，当有新的需求出现时，最初的决定对需求的满足就会变得十分有限。这时，就得修改代码以适应新的需求，但这种修改会违背设计初衷，使得代码变得越来越难以维护。尽管最初设计的是个圆球，但最终它却成了一个扁球，而且还伤痕累累。

大圆球的谬论其实是一种幻想，它认为软件系统的需求在交付之后是不会发生明显改变的，而且，就算改变了，这种改变也是能被控制的。

早期软件工程的研究人员认为，如果在编码开始之前就可以完全明白项目需求，那么就不会存在维护危机了。有些人注意到，问题往往是由于交付后发生的需求变化而导致的，因此这些改变被贴上了邪恶的标签，人们认为固定不变的需求才会产生更稳定的系统。一些人试图限制用户要求改变的权利（例如，詹姆斯·马丁和卡玛·麦克卢尔在 1983 年曾列出很多“针对维护的解决方案”，其中一条就是，“对用户改进加以计划和控制，从而降低对改变维护的需求”。）



令人遗憾的是，这样的严格控制会起反作用，会降低软件系统对最终用户的实用性。正是由于这种只顾眼前利益的做法，20 世纪 90 年代，IT 部门疏远了用户，随后各个公司开始研发那些更小（且通常是复制品）的软件系统。

需求沙堆在我们的脚下不断变化。由于一些理由充足且很简单的原因，软件项目的需求会发生变化。首先，它们是可以改变的。软件是一种具有可塑性的工具。一般而言，改变软件比对硬件做出同等的改变要更符合成本效益。

其次，软件的使用者大多处于竞争的环境中。他们个人之间互相竞争，同其他企业也进行竞争。因为他们要努力去竞争，所以就会转向他们系统中最具可塑性的部分，以寻找新优势。软件的灵活性很诱人。

如果摒弃大圆球谬论，我们就会更乐于接受变化的需求，并能看到软件可塑性的真实面目：一个我们可以控制的巨大优势。需求肯定是会变的。我们必须维护好代码，我们必须插入新需求，尽管这样做会破坏我们的最初设计。就像常言说的，这就是一个功能而不是一个 bug。

只有先改变自己的思维定式，我们才能设计出可适性强的软件。适应性、设计的灵活性以及应需而变应当是所有新软件项目的基石。

# 63. 应对危机



**詹姆斯·格雷厄姆（James Graham）**
PMP
马耳他塔尔艾布安戈市

2009年1月15日下午3:03，西北航空公司的1549航班从纽约拉瓜迪亚机场起飞，这次短暂飞行的目的地是北卡罗来纳州的夏洛特市。

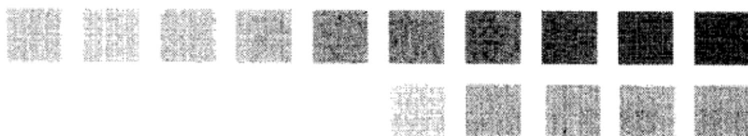
这架空中客车A320载有5名机组成员和150名乘客，机长是切斯利·萨伦伯格三世（Chesley Sullenburger III）。飞机在飞越纽约布鲁克林时遭遇鸟击，两个引擎严重损坏，飞机失去动力。

航空录音磁带中突显了两件事。首先可以感觉到，萨伦伯格迅速意识到发生了不可思议的事情，他的大脑在不停地思索，他多年积累的经验和所受的训练在此时发挥了作用。其次，空中交通管制人员同样迅速做出了反应，他在不干扰机长的情况下不停地给出帮助建议。

在接下来的几分钟里，萨伦伯格意识到飞机不可能安全降落到拉瓜迪亚、纽瓦克或附近的泰特伯勒机场，于是决定在哈德逊河上迫降。他一定是尝试了“延长”飞机滑行，想让飞机降到某个机场，但这位专业机长在权衡了所有可能性的风险后，选择了一个能使机上全体人员安全的方案。

这是一个极好的危机管理实例。

飞行员们在每一个重要飞行阶段前都会讨论飞行计划，并使用能够帮助他们处理正常或非正常事件的备忘录。乘客若能得知这些，一定会感到很安慰。这意味着飞行员们作为一个团队清楚地知道在飞行中将要采取哪些重要行动。



这个危机需要机组人员团队协作，一起工作。当萨伦伯格驾驶飞机时，大副杰弗里·斯凯尔斯（Jeffrey Skyles）努力重启引擎以使飞机在跑道上着陆，而空乘人员在帮助乘客应对水上迫降。西北航空公司的每个人都在为确保好结果而努力着。

说到软件项目，想想你的团队是否能做到以下几点。

- 定期召开团队通气会，并在关键阶段（例如测试）之前，加大通气会的频率。
- 记录每个风险，且有确定适当的应对措施。
- 风险记录定期更新，保证是最新的。
- 团队专家所接受的培训达到相应水平。
- 制定了危机管理计划，关键任务已分配到人。
- 危机管理计划有清晰的内部及外部沟通策略（计划）。

如果你的答案都是肯定的，那就太棒了！你就不会失眠了。但是如果答案不是肯定的，那么明智的做法就是赶紧思考和计划。

为应对危机明晰责任是良好的开端。这项任务可以提前做，其他需要提前完成的任务还包括为每个关键项目阶段准备检查清单、步骤和程序。这些任务都要纳入项目管理计划及详细计划中，并就此沟通，让所有团队成员都清楚。

1549 航班给我们的启示是，一个有能力、有明确职责定位的团队，能够成功应对最具挑战性的危机。

# 64. 了解集成要点



**蒙特·戴维斯（Monte Davis）**
MCSE
美国内布拉斯加州奥马哈市

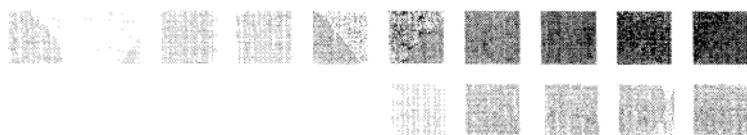
令每一位系统管理员、开发工程师和软件项目经理头痛的事就是系统集成。无论新开发的应用软件、新购买的软件包或是期待已久的新功能升级软件多么有前景，其商业价值都建立在该软件能够在现有公司系统中顺利运行的前提上。

如果你是一位有经验的项目经理，但不熟悉信息技术领域，那么不要被集成这个词迷惑了。集成仅仅是指把所有不同的软件项目结合在一起，使所有子系统一起工作，提供比单个应用程序更多的功能。例如，你希望只输入一次数据，信息就可以顺利进入销售代表系统、应付与应收账款系统等，让不同雇员都得到其想要的信息，不管他们使用的是什么软件界面。

遗憾的是，软件需要升级时总是令人很紧张。这些升级可能给正常的软件运行过程带来麻烦。不久前我们就遭遇了这种情况。我们的一个软件系统按计划升级，在升级过程中，承包商遇到了意想不到的错误。

导致这次升级失败的是几个视图（即用以显示数据库某些字段信息的屏幕显示界面）。负责软件升级的承包商不知道这几个视图的用途，因此把它们删除了。升级的其他过程看起来都很顺利。

几天后，另一个完全独立的系统出现了问题。用户发现上周末刚升级的系统没有提供任何新的客户数据。原来，被删除的几个视图就是客户数据的数据源。



自从为完成升级而删除了数据源视图后，系统间的同步过程就断了。为了解决这个问题，我们不得不花费数小时，可最后却发现问题的原因是数据源视图丢失了。然后，为了让两个系统再次建立联系，我们又不得不用手工重建被删除的几个视图。

大多数 IT 部门都有显示硬件系统各组件连接方式的图表。不过，我们发现用图来展示组织内部数据流也同样有帮助，它可以显示数据从一个应用程序流入其他程序的关键连接。

我们明白了，对于这样的情况，有效的方法是建立良好的文档来说明哪些系统是互相关联的。在初次见到负责我们系统升级的承包商时，我们就可以与他们分享那些商业数据流程图。

早先碰到的升级问题本可以用另一种方式解决，而不是删除支持其他系统的关键视图。我们本来完全可以缩短停工期、减轻管理压力，在升级流程结束后自信满满，不必忧虑给系统带来其他隐藏的问题。



# 65. 分布式项目要积极促进沟通



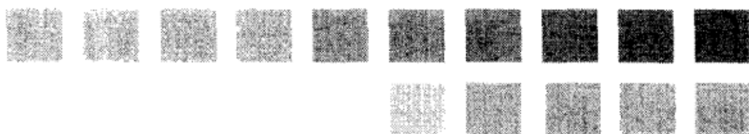
**阿努潘·昆杜（Anupam Kundu）**
美国纽约州纽约市

项目团队成员不在同一地（不真正在一起），导致分布式项目通常会遇到极大挑战。因此，下面这些问题成了项目成功的阻碍。

- 地理位置分散的团队间缺乏信任。
- 用于沟通的时间不宽裕。
- 由于文化差异而不能培养团队整体感。
- 有团队成员缺席例会。
- 项目团队不能和谐共处，因为不同地方的团队成员可能讲不同的语言，或者有不同的做法。

对于很多管理分布式项目的经理，这些绊脚石已经成为了噩梦。如果你被任命为分布式项目的经理，以下几个方法可以加强你的沟通手段。

- 查出不同地理位置的团队有哪些都方便的共同时间（不要忘记夏令时）。
- 公开所有团队成员的即时通信地址（以及能联系到他们的最佳时间）。
- 确保每个关键利益相关者都知道电话会议的接入细节（网络和电话）。
- 汇总各团队的节假日信息，并标注在同一个日历上。
- 公布不同地理位置的团队每天的站立会议安排。站立会议比座谈会议好，参与者会更专注，因为没人希望站很长时间。
- 公开每个团队成员的姓名和大头照，确定每个关键岗位都有一名后备联系人。
- 建立一个公用地址来分享团队间的项目成果（文件、报告或模板）。



除了增强沟通策略外，你还需要注意一些后勤问题，它们有助于提高分布式团队间的高层沟通。

- 所有地点都要有高质量的扩音器。参与电话会议时，确保不同团队之间有稳定的电话连接，对建立同事友谊大有帮助。
- 把电话放在一个有大桌子的宽敞房间里，你要让大家坐得舒适，也许还需要为参加会议的人提供食物。增设一个白板，及时记下电话会议的内容，使屋内的每个人都能够看见。
- 要为一些团队成员预留出差经费，在开始阶段或者质量保证流程中，这些团队成员可能需要去另一个团队所在地。
- 建立一个项目仪表盘（使用任何协作工具），用于团队间沟通问题。在团队间分享仪表盘上的信息，无论他们使用在线工具还是仅仅使用数码相机分享技术。
- 在一个公用地址发布项目的整体目标，确保每个成员甚至在家办公者都能看到。
- 安排商业赞助人做陈述，并坚持让每一个地区的关键成员参与。

随着虚拟团队和分布式团队变得愈来愈常见，只要拥有新颖的沟通技巧，你就会有更多的成功机会。

# 66. 在开始时就要胸有成竹



**路易斯·托雷斯（Luis E.Torres）**
PMP
哥斯达黎加阿拉胡埃拉省圣拉斐尔

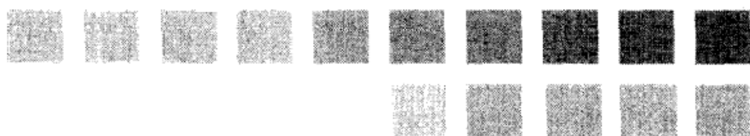
恭喜你！你成为了每个人都心仪已久的软件项目的项目经理。整个公司的希望都寄托在你身上。你的直觉告诉你，赶紧回到办公桌前开始起草项目计划，对吗？不过，为了增加项目成功的机会，你首先需要做几件事情。其中一件就是"在开始时就要胸有成竹"。

首先，看看工作说明书（SOW）、合同或其他文档，弄清楚客户的期望和需求。了解期望和需求的差异（我期望得到一个 SUV，但是我实际需要的是一个小排量的小汽车）。现在，你就能更好地结合"期望"和"需求"，更能够回答下面这些问题："我们要努力完成什么？什么能让这个项目在客户、我的公司和我自己之间实现共赢？取得成功后会得到什么？"

对于最后一个问题，仅仅回答"合理利润"是远远不够的。你还希望客户成为回头客，希望项目团队成员想再一次和你合作，并且希望自己成为标杆人物。

正确的态度和正确的人力资源管理技巧，是你成为成功项目经理的关键。你要与项目团队成员一起召开项目启动会议，回顾工作说明书，对交付物达成共识。

接下来，定义项目范围，建立工作分解结构；确定必须达到的质量参数；制定项目计划；计算所需资金。这些因素（范围、质量、期限和成本）是你应该监督和控制的基本因素，也是你的项目计划的基石。



一旦你把项目分解成了一个个可管理的小块，就必须确认最终产品应该具有哪些特征才能满足项目质量要求。当你适当调整了项目范围，并注意了需要遵守哪些质量规则后，就能轻松地确定这个项目需要多久才能完成。

为了计算完成项目所需的时间，你需要确定完成每项任务的时间、任务之间的相互依存性、具体限制和可用资源。之后要考虑成本，因为它通常与你要做的工作、完成工作需要的时间和资源有关。例如你要聘请一名顾问来完成某项特殊任务，如果计划让那个顾问工作一周，那么和计划让他工作 10 个月相比，花费的成本肯定不一样。最后，还要考虑采购、沟通和人力资源。

在开始时就考虑周全，成竹在胸，你就更有机会获得成功。



# 67. 清晰的条款，长久的友谊



马泰奥·贝基（Matteo Becchi）
PMP
美国弗吉尼亚州阿林顿市

这个标题来自于一句古老的意大利谚语："Patti chiari，amicizia lunga。"意思就是"清晰的条款等于长久的友谊"。

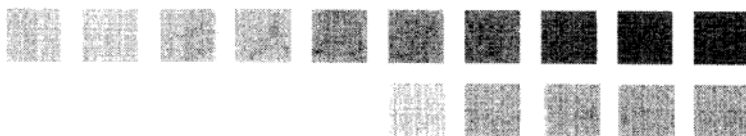
我认为这个说法适用于项目管理的很多方面。从更宽泛的方法论层面看，这句话道出了范围说明、设定目标和可交付物、创建项目定义文档的用意。事实上，所有项目成果都瞄准首先提出项目团队要实现的条款和目标。

现在我们从更高角度来看这个概念。根据 *PMBOK® Guide* 的九大知识领域，从制定项目规章、范围、工作分解结构（WBS）、计划、成本预算，到质量、人力资源、沟通和采购计划，来看看项目生命周期的初始阶段和计划阶段。

所有这些活动都强调，关注焦点应放在提前计划以及就计划与每一个利益相关者进行沟通，确保所有人意见一致。这些基本措施确保了项目顺利度过整个周期。

其次，在战术层面上，当召开会议时，要确保建立（即简单陈述并制定）清晰的项目会议指导原则以及团队的期望，如下所示。

- 事先拟定好具体议程，并准备一份与会人员名单。对于高薪优秀软件开发人员，如果你在会议上浪费时间，就是在浪费他们的编程时间，而且你的预算很快就会超支。
- 大家一致同意，每一个参与者都去收集信息、与外部专家探讨、阅读相关出版物或论文、查阅以前的笔记或公司记录，做相应的准备工作。因为一个团队成员没有做好准备而不得不与高薪人士召开第二次会议，是不能原谅的过错。



- 如果需要接入笔记本电脑、设置投影设备或者接通用于演示的自动设备，那么你就要早些到达开会地点做好准备工作。如果不需要这些，也要早到几分钟，找把椅子，喝点咖啡，与其他团队成员打个招呼。
- 规定开会期间不得使用通信设备（黑莓、笔记本电脑和移动电话）。如果你已经开始讲话了，而那些技术高超的开发人员正在写东西或者玩游戏，那就说明这些编程人员在走神。
- 尊重其他的项目团队成员，当有人说话时，不要窃窃私语、插话或者干扰正在发言的人。

再次，和客户、承包商和转包商签订清晰的合同。销售部门仅仅关注要交付的最终产品和公司能接受的最终价格。合同中要明确哪些具体情况下可要求作更改，以及对这种改变如何收费用。

根据项目具体要求，努力建立一套流程，规定与客户沟通的频率和形式。有问题时能联络到你的客户吗？当研发取得进展时，他们愿意提供最终使用者来测试软件功能吗？

清晰的条款等于长久的友谊——无论是什么项目环境，这句话都适用。

# 68. 做实际工作的人才是最好的估算人员



**乔·泽尼维奇（Joe Zenevitch）**
美国纽约州纽约市

你参与过只有一个人估算所有工作的项目吗？这种方式成功了吗？我猜，很可能没成功。

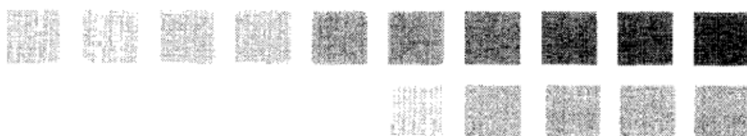
这种方式存在如下三个主要问题。

- 除非你很幸运，否则团队开发人员和估算人员的技术水平是不会在同一层次上的。因此，就算是由主要架构师做所有的估算工作能确保结果准确，但是研发人员的步调肯定会千差万别。
- 一个人为整个团队做所有估算时，出现错误的风险会非常高。越多的人参与估算，结果就越好。
- 研发人员要被动地接受一个他们必须实现的估算值。我很少看到研发人员会喜欢这种方式。

软件项目经理最大的错误就是认为自己有资格帮助整个团队做估算。他可能会认为，自己曾是一名研发人员，能恰当地选择估算值。但即使现在他仍在做研发工作，也请考虑再三。主要架构师也可能会有这种同项目经理一样的想法。但是，你做实际开发工作的时间越久远，你进行估算就会越糟。而且如果团队使用的是你所不熟悉的技术，那就更不要考虑估算了。

在我们的项目中，我们使用宽带 delphi 法进行小组估算。我们先让业务分析师描述对某项功能的需求，研发团队听，然后团队成员再提问以解答所有疑惑。

一旦他们准备好做出最初的估算，他们每人会在卡片上写出自己的估算值。完成后，数到 3，所有人都亮出卡片。



现在，我们就可以比比这些估算值了。如果它们非常接近，我们就选取较保守的估算。如果存在很大差距，那么我们就要求研发人员对自己的估算前提条件进行讨论。经过多次讨论，我们要求他们再做一次估算。这时常常发生的情况就是，当研发人员在完成这个功能所需的条件上达成共识并取得一致意见后，估算结果就会逐渐集中到一个数字上。

这种方法极具优势，原因如下。

- 整个团队都参与到了估算活动中，并且分享所有不同的观点。团队成员常常会团结一心，很快得出一个共同的估算结果。
- 之后，当实际的编程工作开始时，由于研发人员都了解得出估算结果的思维过程，就不太会出现只有一小部分人能做某个功能的情况。
- 有了团队自己的估算结果，就不太可能有反对意见了。就算他们的估算可能是错的，团队成员也并不会那么抵触，而且为了修订估算会更加团结协作。

记住，最好的估算人员是那些做实际工作的人。



# 69. 沟通最关键



**杰纳迪·米罗诺夫（Gennady Mironov）**
CPM
加拿大安大略省多伦多市

在任何行业，项目经理应该掌握的最关键的知识都是，如何成为一名优秀的沟通者。一个人可能拿到了很多认证证书，拥有一串头衔和委任职务，但是如果他不了解如何与他人合作的基本知识，就不能很好地完成项目工作。

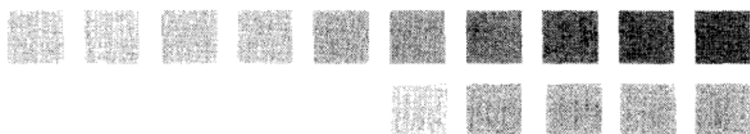
我坚信，当一个新的项目开始时，优秀的项目经理应该与所有的利益相关者面谈，特别是客户。项目经理要介绍自己，讨论项目目标和所有重要问题。如果利益相关者、客户甚至项目团队不是身处五湖四海，那么面谈不会是个大问题，就算经济低迷。

有句谚语："百闻不如一见。"根据我管理数百万美元项目获得的经验，我发现若亲自拜访客户，很多问题都能在半天之内轻而易举地解决。

有一个项目，承包商没有为我们提供所需要的无线基站，致使项目进度延误。这个问题在于，承包商把他的那份项目工作转包给了分包商，而这个分包商迟迟不能提供电力系统。尽管我们花费了数周时间打电话，往返了无数封邮件，但问题还是没有解决。

最终，我决定去见见承包商，解释一下问题的原委，提出一些可能的解决办法，于是我们得到了所需的设备。在大多数情况下，如果你愿意倾听并且帮助寻找解决问题的合理方法，客户是会站在你这边并且随时支持你的。

还有一个项目，客户坚持要求缩短的项目时间。他想在年末压缩设备的生产周期，而那时候所有的工厂都在开足马力完成尽可能多的订单。我们不能接受这个要求，因为它比标准项目时间少了一半。



于是，我组织我们公司、客户和承包商进行了三方会谈。我们坦率地提出了一个最短、最实际的计划，并且详细解释了为什么不能再缩短时间。我们进行了一次特别的盘库，看看这个项目需要的东西已有多少存货，然后，我们冒着一定的风险，甚至没有收到客户的订单确认，就开始了生产。

我们的项目完成得很成功，而且面对这么紧的计划还提前了两天完成。我们的客户非常高兴，新年一开始就为我们提供了另外一个两百万美元的项目，这让我们意想不到。我们不但达到了进度、范围、预算和质量的所有要求，而且还为公司赢得了其他好处。

软件项目依赖于面对面的沟通。



# 70. 项目就是对解决方案的追求



**辛西娅·伯格（Cynthia A. Berg）**
在读博士，PMP
美国亚利桑那州格伦戴尔市

作家斯蒂芬·科维曾写过“在开始时就要胸有成竹”。除了找到最终解决方案，项目还能是什么？要想构想好软件项目的结果，最好的方式就是创建工作分解结构（WBS）。WBS 在层次上将项目的全部范围细分为多个交付物[①]，它很像是组织结构图，将公司分为多个部门、工作团队。然后，再将交付物分成更小的组成部分，直到分成工作包[②]。

制作 WBS 时，团队、资助人和其他利益相关者都要参与其中，这能确保完全确定项目工作，并体现所有参与者的需求。为什么要包括团队呢？显然，有谁会比真正操作项目任务的团队成员更了解需要做的工作呢？当项目经理认为只有他一个人了解怎样细数项目工作的方方面面时，项目就注定会失败。

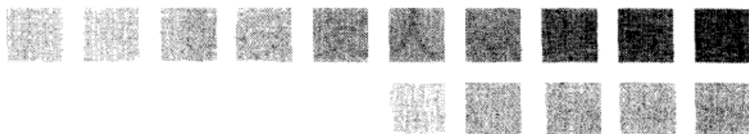
在建立 WBS 时，团队就有可能挑战既有的做法。而且，团队成员可以就项目工作的构成部分达成共识。这种方法能确保他们的努力得到更大认同。毕竟，为自己帮助设计的项目工作总是更有意思。

WBS 中的具体工作任务要分解到多小呢？这是有技巧的。WBS 中没有显示具体的工作任务，因为它只分解到工作包层次。一旦将工作包分配给部门、小组、承包商或者次级承包商去完成，它就有可能再进一步细分为能确保项目高效高质完成的具体工作任务和里程碑。每一个接受工作包（分级最低的工作）的人都应该根据主进度制定他自己的项目计划。

---

① 交付物：通过项目的具体实施而得到的产品、结果或执行某项服务的能力。

② 工作包：交付物细分的最小组成部分，包括具体工作任务和计划里程碑。计划工作包的目的就是将这项任务分配给某个人、小组或承包商。



这时，WBS 就成了项目中所有其他计划、执行、监督和控制职责的支柱，同时，它也可以作为这个项目内外部之间的简洁交流工具。WBS 图示就是一幅项目解决方案图。一旦完成了这个图，就可以开始制作详细的计划、进度和预算。如果不能清楚地定义项目工作，如何能制定计划、预算以及进度呢？

WBS 也是个很有价值的头脑风暴工具。有了展示整个项目的图示，就很容易发现遗漏和多余的部分，或者找出易于提高项目价值的丰富的工作包。要识别（内在和外在的）潜在风险，也可以查看 WBS 的每个部分。

事先制作一个清晰的 WBS，并予以布置，加以解释，让所有利益相关者就此达成共识，是成就优秀项目的秘诀。



# 71. 傻瓜，人最关键



**阿德里安·威布尔（Adrian Wible）**
美国纽约州纽约市

永远不要忘记，你的项目团队成员都是有野心、有优势、有局限并有弱点的人。项目的成功更多依赖于团队成员的态度和才能，而不是你自己做图表的高超技能和项目跟踪的超凡能力。管理项目可以很随意，但是不要忘记领导团队。

大多数项目经理都处于矩阵式管理[①]环境中，其团队成员同时向项目经理和部门经理做汇报。对于他们，项目经理并不肩负人力资源的聘用、开除或评估的职责。然而，项目经理不能放弃关心和支持团队成员的责任，而将这些责任一味留给人力资源经理或其他职能负责人。

大部分经理都是因其人力资源或专业知识而得以晋升的，而不是因他们具有激励员工的能力。项目的成功取决于你的领导能力。有很多书讲述领导力，你应该如饥似渴地去阅读。

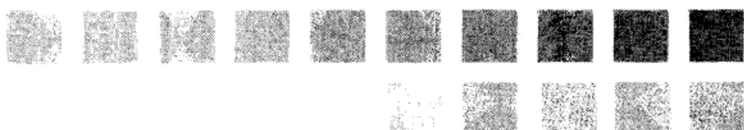
团队中的每个成员都希望多出力、多学习并有所成就。有时，深入发掘一些团队成员的这种愿望也是一种挑战，而这就是软件项目管理富有挑战性和充满乐趣之所在。

要经常和你的团队成员一对一地交流。发现他们的问题，征求他们的想法，让他们在项目中享有发言权。认真对待他们的提议并就提议采取行动。

问问你的团队成员，成长之后他们想做什么。态度要诚恳。我们都有工作志向。做一位关心他们事业的良师益友，由此激发出的力量会令你惊讶。

---

① 矩阵式管理通常将管理部门分为两种，一种是传统的职能部门，另一种是为完成某一专门任务而由各职能部门派人联合组成的专门小组，并指定专门负责人，任务完成后，该小组成员各自返回原部门。——译者注



对待团队成员，要开诚布公、坦诚且直率。定期给出反馈，不要仅仅在复查阶段。反馈应对事不对人。再强调一遍，管理艺术很丰富，要不断学习。

一位团队成员工作做得不好时，你可以运用CRAM模型：限制（Constraints）、资源（Resources）、能力（Aptitude）和动机（Motivation）。项目经理经常将表现不佳视为动机问题，但CRAM模型建议动机应是最后考虑的因素。团队成员在生活中可能遇到制约他工作效率的各种因素，例如离婚或结婚、有孩子、克服成瘾问题等。

团队成员可能无法拥有以最高水平发挥其能力的资源。例如，没有质量保证（QA）测试环境或者硬件太陈旧；预算不足使其无法建立测试环境或者购买必要软件；也许接触不到领域专家（业务分析师、客户、最终用户）。

你的团队成员也可能无法胜任目前的工作职责。他可能没有完成这个项目应具备的编程能力。如果是这样，尽可能在项目中为他换一个职位，或者，为他找到另一个能发挥其长处的团队。

当团队成员表现不佳时，动机应是最后触及的因素。只有在考虑了限制、资源和能力之后，才应提出动机问题。

做一个领导者，与你团队的每个人紧密联系，这样做会让你喜出望外。

# 72. 文档是手段而非目的



**帕特里克·夸（Patrick Kua）**
英国伦敦

艾森豪威尔曾经说过：“计划是没有作用的，但是计划过程却是不可或缺的。”成功的项目经理都懂得如何从计划过程中获益，而不会过分频繁地更新计划。他们积极利用文档发起有意义的对话，而不会用它来替代所有的沟通方式，更不会在人们违约时用文档来指出这种违约行为。

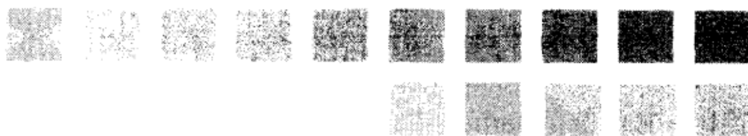
对于项目经理来说，计划部署和跟踪始终是重要的工作任务，虽然这些任务都是为了实现某个目标。很多公司错误地衡量项目经理的工作，他们关注的是项目经理是否坚持计划，或者完成、分发、归档文件有多彻底。

在错误理解计划的公司中，项目经理会被问道：“你完成计划的准确度有多高？”如果被问及这种以微观管理为中心的问题，而不是寻问诸如“你在要求期限内交付了最有价值的任务了吗？”之类更重要的问题，这样的企业你一定要小心。是否有价值应当看是否在规定的预算内完成了正确的目标、让客户满意甚至超出预期。如果手握错误的准绳，那么很容易就会忘记真正的终极目标是什么。

仅仅关注制作计划、弄出一套完美的文档，会在进展和实现方面制造假象。这意味着计划的实施很容易，计划很精确，但事实却并非如此。

我曾经见过，项目经理试图强迫每个参与项目的成员严格按最初计划规定的工作任务和时间开展工作。他们没有意识到，当条件发生变化时，根据新的形势重新计划工作对带领团队更有益处。

对于企业来说，计划和文档都包含了完成目标的重要信息。但是，计划和文档本身的确没有什么用处。要想发挥它们的作用，人们应该依据计划和文档所强调的结果来工作，并且将它们所包含的信息传递给其他人，以使大家从中受益。



因此，哪类信息需要传递，向关心项目结果的其他参与人传递信息的最佳方法是什么，这些才是始终需要重点考虑的问题。要想传递重要数据，文档通常是最不应该选择的方式。最佳的沟通方式是面对面的交流。

项目经理还要肩负一项不轻松的工作，既要满足可追溯性或审计要求编制文档，又要尽量多开展那些不以文档为中心、最终能增加项目价值的活动。

成功的项目经理仅仅做够用的计划，把握够用的细节，认识到随项目的进展会不可避免地出现的问题，并且知道什么时候计划需要因新的或始料不及的需求而改变。要时刻牢记，计划流程中的文档是使项目正常运作的手段，而不是最终的目的。



# 73. 报告中挣值与速度两种度量能共存吗



**芭比·戴维斯（Barbee Davis）**
文学硕士，PHR，PMP
美国内布拉斯加州奥马哈市

软件开发者们越来越确信，要想创建高品质且能解决客户问题并带来商业价值的实用功能软件，最佳方法是采用更加敏捷、灵活的方式。但是，项目管理办公室（PMO）却在继续使用较为传统的方法制定流程、培训项目经理，这些传统方法在 IT 之外的许多领域是成功的。

为了让高层管理人员获得适用于这两方面的标准，有没有什么办法在报告中融合这两部分？有办法。可能行得通。

先说挣值（earned value），它就是用数据说话，跟踪项目进程以及进程中每周、每月或每季度商业价值。简单来说就是，忽略成本因素，由项目经理（和其他利益相关者）定义需求并且估算这个项目需要花费的时间。再将这些预估值转化成进度。

我们假设汇报周期是一周，项目小组估计在一周之内能够完成 40 项预定任务。周五下午，小组汇报他们的实际进展。如果小组在这 40 个小时内完成了所有的任务，它就挣得了 40 个小时的价值（EV）。最初预估或者计划的 40 个小时的价值为计划值（PV）。EV−PV=SV（进度差）。本例中，这个团队的进度差是零。

但是，如果团队落后了，进度就会落后，那么与这个进度有关的后续其他工作人员就需要做出调整。如果团队提前完成了任务，那么原来的估算就多了，投入的物料或其他项目参与人员需要了解工作可能会早于预期开始进行。记住，项目的范围（工作）早就被设定好了。



而在敏捷开发方法里，用“速度”这一术语来衡量开发者的生产能力。我们允许人们在下周承担的工作量预估算值不会超过他们上周完成的工作量。然而，由于这个开发者只是与自己以及他上周的选择进行比较，而不是与一个长期的计划相比，所以就没有必要重新安排其他人的工作。而且，这周的工作也许很简单，bug 更少，或者是程序员已很熟悉的任务。

在软件开发项目中，最终产品的功能并没有完全固定下来。所以，如果速度并不像原来预计得那么快，范围（交付功能的数量）就可能缩水。

由于市场、生产和培训等问题，软件开发项目中会产生大量的报告，而软件项目经理整日要应付这些报告，因此需要统一报告的度量标准。最简单的方法就是给 IT 团队一段时间（和相应的人力）来处理该软件，例如，在报告中展示 5 周的时间。当你的 IT 团队递交每周软件报告时，也让他们一同递交完成了的功能或故事，你再将之转换成任务名称，事后放入这份报告。现在，这些任务才可以更新，显示已经按照计划 100% 地完成了。这种方法可以在传统的报告中展示敏捷开发的进度。



# 74. 范围改变经常发生，要适应它

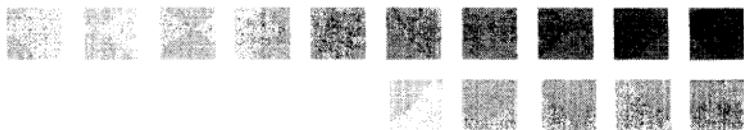


**帕维尔·西姆沙（Pavel Simsa）**
PMP
美国华盛顿州贝尔维尤市

如果说有一种东西是软件研发项目特有的，那就是不可避免的范围改变是如何发生的。并不是说其他领域的项目就一定不会发生范围改变，但是我认为，其他任何行业都不会有如此频繁的范围改变。

我们知道项目受到三方面的限制：成本、时间和范围。

- **成本**。如果项目遇到麻烦，此时投入再多的资金或资源都无济于事。如果是挖战壕，你将挖掘者的数量加倍，可能会在略多于一半的时间里挖好战壕。但是，如果你希望软件项目重回正轨，而将开发人员的数量增加了一倍，那么很可能会适得其反。你会给那些知道如何编码和如何做事的人制造巨大的混乱。所以，成本需要保持不变。
- **时间**。总是有“日期”。它不是最初进度表上标明的交付日期。虽然没有人明说，但是，如果你在开发一个大型安全产品，原计划于 11 月交付，很有可能交付日期会一直延迟到 1 月，而届时你们还在开发不止。秘而不宣地，团队成员都知道最后交付日期会就在 2 月，例如“在国际黑帽安全会议期间，届时将宣布新产品发布的消息”。你在交付时间上有一些灵活性，但很小。时间是限定的。
- **范围**。不断发生变化的是范围。很奇怪的是，范围是一个最善变的制约因素，特别是当开发商业软件而不是为了特定客户定制软件的时候。原因很简单。每一个新的软件产品都有“必须具有”和“最好具有”的特征和功能。通常来说，“最好具有”的功能的数量要成倍于“必须具有”的功能。



幸运的是，“最好具有”的功能也最容易被消除。如果你正在建一座摩天大楼，你不能在项目进展的中间阶段宣布：“为了让这个项目回到正轨上，我们就建造 40 层，而不是按计划盖 60 层。等我们以后有时间，可以再加盖 20 层。”

对于软件，就会很容易说：“计划变了，第一个发布中，我们将只支持两个操作系统。之后，我们可以增加另外两个原计划支持的系统。”

这并不是完美的解决方案，我们能采取什么措施避免发生这样的情况呢？老实讲，很可能没有。这就是软件开发项目的本质。然而，你能做的就是具体地计划项目范围。识别出“最好具有”的功能，以及它们一开始就有的相关性。相关性很重要。删除“最好具有”的功能时可能会同时改变与“必须具有”的功能相关的开发架构。

如果你在开始时就计划项目范围可能会缩减，那么在有必要缩减范围时，你就会很容易决定缩减什么以及如何缩减。



# 75. 买现成的软件



**埃尔纳尼·马奎斯·达席尔瓦（Ernani Marques da Silva）**
MBA，PMP，PgMP
巴西圣保罗州迈里波朗市

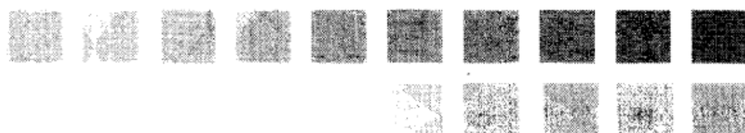
现在，购买现成的软件（可以立即进行测试、实现和使用）很常见，也很实用。为什么呢？这样的软件能够使公司提高效率，并通过减少花费在软件开发和实现阶段的时间来优化其效率。这样的购买，你买的不仅仅是软件，还有写这套软件的公司的技术。

当然，每个公司都有自己的流程、政策和遗留软件（例如，会计系统、安全软件等）。因此，为了适应公司的政策和流程，并与先前安装的遗留系统集成，新的软件通常需要定制。在大多数情况下，销售商会在预售阶段就了解定制信息。

这里就是可能出现重大问题的地方。即使你遵循的是非常详细的采购流程，要想确定新应用程序里固有的某些功能（例如，公式、数据输入显示、与遗留软件集成等）是否按预想的商业（产品）领域要求工作，也是很复杂的一件事。一旦完成采购流程、签订合同、同意并执行项目计划，问题就会在测试阶段出现。问题可能与定制有关，或者在最坏的情况下，与根据演示而被认定的软件功能有关。

在签订合同之前，一定要遵从下列几个步骤。

(1) 就公司的软件需求准备一份非常详细的清单。
(2) 访问公司，并且准备一份详尽的调查报告。
(3) 准备卖方评估报告、测试案例和测试计划。
(4) 确保完成测试案例，并有文档记录。
(5) 在签订合同之前遵循测试计划（案例）。



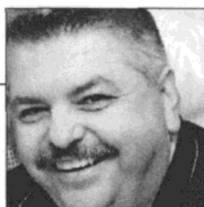
差距以及填补这些差距的计划应该得到双方公司的理解和认可。在认真执行并用文档记录了这个流程之后，你就会获得明确的信息，以决定购买哪种软件、估算定制软件所需的时间，以及与之相关的实际成本。从长远来看，你将因此节省时间和资金。

这听起来好像在选择卖方和签订合同之前我们要花费很多时间。没错，现在投入这些时间远远胜于坐等软件到手再安装。如果在测试阶段（甚至软件已经交付给最终用户后）才发现太多不兼容的问题，那么成本就会迅速猛涨。

重述一下，如果你的公司决定要购买现成的软件，在购买之前，一定要多用一些时间去分辨真正的需求，并研究准备购买的软件在功能和技术上的细节。无论软件供应商对你来说是熟人还是陌生人，无论软件是影响较小的台式机应用软件还是对公司至关重要的服务器应用软件，你都应该使用这种方法。



# 76. 三类项目赞助人



乔治·格拉伯特（Jorge Gelabert）
PMP
美国康涅狄格州柏林市

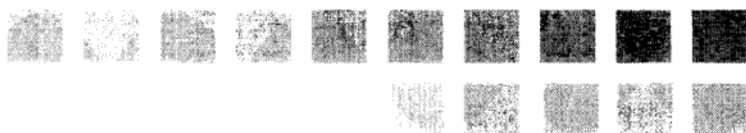
每个项目都需要赞助人。项目赞助人通常就是项目的发起者，以及为项目成功完成提供资金保障的人。通常，赞助人也是组织中具有很高地位的人，他会支持这个项目，并且在项目经理遇到无法处理的问题时参与其中。项目越大，一个有力赞助人的重要性就越大。

以我的经验来看，大概有三类赞助人：好的、坏的和恶劣的。识别每一类赞助人并了解如何应对，是非常重要的。

最差的赞助人就是“恶劣”型的。这些赞助人通常是被指定的，因此，在将要交付或打算使用的项目中，他们没有个人投资。这样的赞助人往往不愿听从于项目经理，只会关注分配者武断设定的项目完成日期。这种并无恶意的忽视是很常见的。指定的项目赞助人可能会频繁更换，因此没有连续性。

识别这类赞助关系很简单，但要解决这个问题却不容易。软件项目经理必须与赞助人一起工作，并且满足他的需求。这经常与成功完成项目相矛盾。有个办法就是找一个代理赞助人，它可以是一个人或一个团队。这个代理赞助人既能从项目的交付物中获利，又能为项目经理提供赞助人所能提供的帮助。或者，项目经理可以要求其他人代表自己对现有赞助人施加影响。作为项目经理，你的成功在很大程度上依赖于你在公司内部的协调工作做得有多好。

“坏”赞助人可能会以不同方式阻碍项目的进展。他可能会参与到通常应由项目经理处理的日常事务中，和团队成员直接沟通，做一些不合适的项目决定，篡夺项目经理的权力并扰乱团队。作为赞助人，他可能能力很差，不能提供所需的资源，受其他事情拖累，或者没有时间为项目提供指导。



防止出现“坏”赞助人的方法就是清晰界定赞助人应有的职责和任务。对于喜欢指手划脚的赞助人，给他们一份“工作描述”，以此来规范自身行为。对于能力差的赞助人，要使他们明白你期待他们做什么，让他们意识到自己不能胜任这个角色，那你就可能获得一个更好的项目指定赞助人。

最好的情况就是碰上一个“好”赞助人。好的赞助人明白他们的角色、职责和应有的行为。他们是项目资助者，他们提供资源，在需要时提供帮助，并且支持项目经理的决定。他是一个对项目成功全心投入的公司领导。

不论赞助人是“好”、“坏”或者“恶劣”，就像管理项目一样，管理赞助人同样是项目经理的责任。和赞助人保持良好的信息沟通，仅在必要时才让他们参与进来，同时避免赞助人控制这个项目。要学会识别不同类型的赞助人，并且准备好相应的对策。



# 77. 该少于承诺还是多交付



**乔·泽尼维奇（Joe Zenevitch）**
美国纽约州纽约市

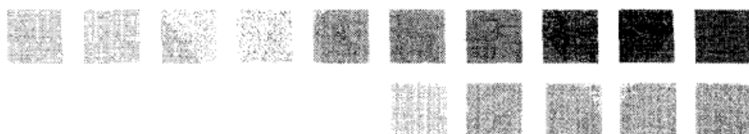
在项目收尾阶段，如果交付的任务少于你之前承诺的，那么你就不是个好的软件项目经理。如果交付的任务多于你之前承诺的，那么你就是个英雄。但实际上，你应该努力交付之前所保证的，既不多也不少。

新项目经理往往急于取悦他人，会让业务人士（客户）不断增加项目功能，甚至是在团队完成交付任务的能力下降时也这么干。商业人士认为项目经理完全掌控了局面，他们会利用这个机会不断增加大量新功能。

由于害怕被看出弱点，这些新项目经理会咬紧牙关，希望自己能够完成交付。但是，当项目临近尾声时，他们就会清楚地发现，那些列在单子上的功能不可能全部完成。于是，项目的删减过程勉强开始了，但删减的功能并不一定是后来增加的。那个开始很高兴的业务人士现在咬牙切齿，正准备解雇这个项目经理，或者等项目结束后再对项目经理秋后算账。

有经验的项目经理都知道，从第一天开始他们就要表现得很坚定。对于增加新功能或改变项目范围这样的事情，项目经理都应该予以回绝。他应该提醒业务所有人，每次发布只能容纳这么多功能。如果提出新功能，那么这个新功能必须推迟到以后的发布中，或者替代原有计划中的某个功能。

有经验的项目经理会避免"高-中-低"的优先等级分类方式，因为客户可能会将每件事情都标成"高"。他们往往依据商业价值对功能进行优先级排序。一个好的项目经理会提醒业务所有人，如果遇到延期的情况，列在最后的功能可能不会在这次发布中完成。这些规则可能会惹恼业务所有人，特别是那些不明白强迫加入功能没有用的人。久而久之，他们会习惯这个过程，并且逐渐接受这就是项目生命周期中的事实。



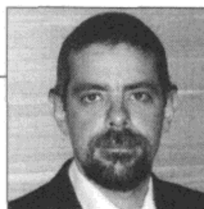
当然，有经验的项目经理能预料到项目过程中会发生变化，并且会将处理偶发事件的时间列入计划。项目经理对待这类时间会很谨慎，有些时候不会告诉客户甚至团队。就像管理稀有资源一样，只有经过“回绝战斗”减不掉的功能才使用这个时间。

这种情况下，业务所有人通常会感谢项目经理最终帮了他一个大忙。在研发的最后几天里，如果还有处理偶发事件的时间，项目经理甚至可能选择“后备功能”，开发一些额外的功能。一些业务所有人可能会质疑，为什么他们不能早点开发这些功能，但是大多数情况下，他们会很高兴接受一些额外的意想不到的东西。

现在，项目经理处在产品发布的最后阶段。团队已经按照承诺完成了交付，有时候还提供了额外的功能。业务所有人很高兴，团队同样很高兴，并且项目经理的声望完好无损。就请在功能发布即将到来时开始狂欢吧。



# 78. 每个项目经理都是合同管理者



**法比奥·特谢拉·德梅洛（Fabio Teixeira de Melo）**
PMP
墨西哥韦拉克鲁斯州夸察夸尔科斯市

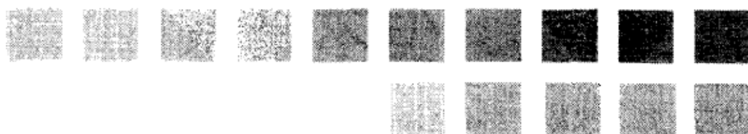
作为项目经理，你有责任控制变化。你要创建一个流程来归档整理需求，并执行这些变化。但是，你若根本不知道发生变化了，又如何能控制变化呢？

客户的团队成员会直接联系你团队中的相应成员。为了让客户满意，或者由于没有意识到合同约定的义务，你的团队成员可能会同意提供额外的培训，甚至实现某个软件变化，而且还忘记了通知你，或者在最后关头才提醒你。在这些改变中，有一些可能没有什么妨害，但是有些可能会引发问题。例如，悄悄改变部分功能意味着可能在软件手册中未提到这一变化，从而导致重写、重印等，并产生与之相关的（不能计入账单的）时间和成本。

有的人可能忍不住想要禁止客户成员和承包商项目成员之间的联系，但是这会危害双方的沟通。合同里也不会约定客户是否有权利和你的团队成员交谈。对于团队成员和客户之间是否应该进行交流的问题，项目经理应该如何处理呢？

为了避免执行未记录的变化，每个团队成员应该熟知合同内容，包括范围、时间和各方的权利与义务。他们在了解合同时就要分析客户的需求，并且应该知道变化发生时要如何通知每个人。这就要求变化控制和处理流程都应该有文件记录，并且团队成员应该熟知这个流程。

研讨会形式非常有效，能够让团队成员了解到合同中最重要的方面和变化控制流程。在项目开始时就要举行这种培训会议。还有一种方法，你可以在内部的项目启动会议上涉及这个主题。无论用哪种方法，你都应该确保每位项目成员都了解这方面的内容。



你要特别关注参与项目的第三方，如提供商、供货商和分包商。控制客户与他们接触，既困难又有些微妙，而且有时候，他们可能和你的客户有着独立的商业关系。一个分包商接受客户需求，实现它，并把账单发给你这个主承包商，这是很常见的。

处理这种问题的最好办法就是杜绝它的出现。与你的提供商谈谈，建议他们告知与项目相关的人你与他们以及客户间的合同关系，同时也将他们纳入变化控制流程中。

记住：项目的成功主要以客户的满意度来衡量。关键在于不要抵制变化，而要控制变化。就这一点而言，你的团队成员所要做的就是发现潜在的变化，并通过变化控制流程来通知你。这样一来，你既可以控制与客户之间的关系，又不必牺牲时间和成本来满足他们的需求。



# 79. 重要，但不紧急



**亚历克斯·米勒（Alex Miller）**
美国密苏里州巴尔温市

斯蒂芬·科维写了关于个人工作效率的经典之作《高效能人士的七个习惯》，那本书里依据纵坐标（重要性）和横坐标（紧急性）来划分任务，有如下四种组合。

(1) 重要且紧急：迅猛龙[1]式的出击。

(2) 重要但不紧急：准备好未来的产品策略，重做产品中有问题的部分。

(3) 不重要但紧急：邻居打电话说借点糖。

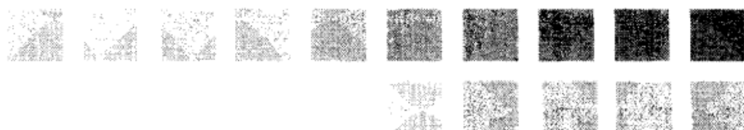
(4) 不重要也不紧急：上 YouTube 网站浏览网页。

我们来一起讨论如何使效率最大化。

先考虑既不重要也不紧急的任务（第 4 种）。大部分这样的任务（你可能将其归类于“不着急处理的事”）可以直接去掉。根据定义，这些任务既不重要（所以为什么还要做呢？）也不紧急（所以可以等一等）。如果第 4 种任务交由你的公司来做，你就应该问问老板，为什么一定要做这些事情。聪明的经理不会让员工做不重要的事情。

你同样要努力减少那些不重要但很紧急的任务（第 3 种）。一种方法就是让事件的发起人与你联系，你依据自己的时间来处理这个事件。另一种方法就是改变你的环境，避免自己被他人打扰。电话和电子邮件通常视为紧急沟通方式，无论内容重要与否。使用语音邮件或者电子邮件过滤器可以降低它们的紧急程度。

① 迅猛龙：一种体格较小的食肉性恐龙，在史蒂文·斯皮尔伯格的科幻影片《侏罗纪公园》中，这种恐龙被认为对人类有巨大威胁。



重要且紧急的任务（第 1 种）通常在出现时就要及时处理。然而，你应该努力建立风险预防体系来消除这些事件的起因。例如，bug 导致了系统在生产过程中崩溃，你就应该分析这些 bug 产生的原因，并且建立质量控制系统防止这些 bug 再次出现。要想减少重要且紧急的事情，最好的办法就是建立反馈循环，以处理这些事件发生的根本原因。

重要但不紧急的任务（第 2 种）是你工作中要做的最重要的事情，这正是你利用知识工作并产生价值的所在。如果这样的任务你能够多完成 25%，你的老板就会给你加薪。

作为软件项目经理，你要让整个团队关注重要但并不紧急的任务。你的工作就是避免团队做那些毫无意义的任务（第 4 种）以及来自其他团队的紧急但不重要的任务（第 3 种）。你作为经理，有权拒绝这些要求。要么你自己完成这些任务，要么雇个仆人去做，或者直接拒绝！

你的团队不能忽视重要且紧急的任务（第 1 种），但是，如果你的团队将所有时间都用于灭火，那么，你就需要去修理引起这些火灾的故障线路（第 2 种）。你可能不会立即看到这么做的好处，但是，久而久之，你的团队花费在重要但不紧急的任务上的时间就会越来越多，这类任务才是项目成败的关键。

# 80. 讲授流程



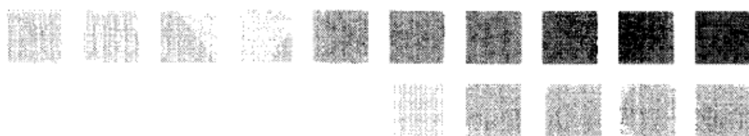
**理查德·谢里登（Richard Sheridan）**
美国密歇根州安阿伯市

一个流程要真正有效率，所有利益相关者就一定要对这个流程有共同的理解。要确保自己的组织做到这一点，一个方法就是对这个项目涉及的所有利益相关者进行正式培训，讲授的主题就是流程。利益相关者包括：项目赞助人、可能有的一些关键用户、项目经理、开发人员、设计人员和质量保证人员。并且，要把他们凑在一起讲授这个流程。

执行客户的项目时，我们要求客户也要接受我们的流程培训。为什么？我们希望项目的赞助人能够理解如何用一种有效的方式来引导团队。我们用通俗易懂的敏捷流程来反对不现实的期望，敏捷流程需要每周估算、计划、展示和告知。

赞助人学习了我们的预估方法，他们就知道如何对待我们的估算值（估算值不是定价投标）。他们学会了简单的计划技巧，就会在这些估算基础上选择范围，并且从商业价值角度进行交叉检查。他们会积极地参加每周展示会，这样一旦有误解就能及时暴露出来。

有一次，我在讲授流程培训时，提了课堂上的两个开发人员的名字来解释一下有关估算的问责规则，我对他们说："特德和基利，你们在公司里永远不会因达不到估算目标而受惩罚。"然后，我又对班里的项目经理说："丽莎，你知道，当开发人员达不到估算目标时，不能给他们压力或者惩罚他们。"然后我又对教室里的买方客户解释道："珍，你知道吗，如果我们超出估算的话，你就得为这项工作付出更多？"



当然，此时此刻，这两个开发人员会认为我在和他们开玩笑，而客户大概准备要取消项目。然后，我解释了估算问责的最后一条规则："基利，特德，我只需要你们两个人做一件事，那就是，只要你们认为达不到估算要求，那就说出来并且告诉项目经理。而丽莎，作为软件项目经理，你应该和他们讨论一下这个任务，确保自从上次估算后他们就没有改变过任务范围。如果确实达不到预算要求，你就必须联系客户，询问他们这时想怎么办。"

最后，我转向客户说道："珍，这下你了解了吧。估算越大胆，就有越多的工作要在更少的时间内完成，因为估算环境是让人轻信的，而且团队成员会非常专注地去努力实现自己的估算，他们乐此不疲。但是，你必须坦然接受这一点，我们隔三差五会犯点儿错误。当然，犯错误时，我们一定会通知你，而不会等到你把所有的钱都花光。"

这种流程培训真是一个非常有效的加强控制的工具！



# 81. 状态的假象



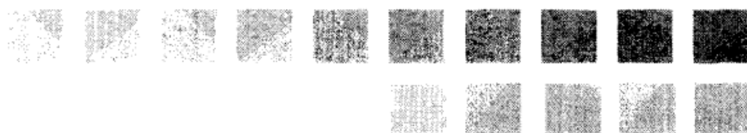
**尤迪·达罕（Udi Dahan）**
*以色列海法市*

在成功完成了第一个项目后，我充满自信地着手第二个项目。这是一个更大的项目，对于单位来说更具有战略意义，我也将要管理一个多学科背景综合团队。我确信在第一个项目中所使用的技巧这一次不会失败。有趣的是，正是由于相信团队的状态报告而导致了我的最终失败。

项目开始了大概两个月后，我的基础架构团队组长坦白说："事实证明，我们假设的架构条件有些是没有根据的。"但是，他对我保证说："我们会在月底之前回到正轨上来。"尽管他再三保证，而且我也准备了应急措施，但仍然有种感觉在我心头挥之不去——一定有地方出了问题。

到月底的时候，我追问那位组长。他向我说明重构工作是如何按计划完成的，所有的开发人员正准备完成他们下个月的目标。然后我和集成团队组长一起坐下来聊时，她告诉我说，从她的角度来看，也是一切正常。模块正照着各自的要求来完成，每一个模块都经过了大量的测试，并且测试表明，各层架构都很稳定，可以做第一次集成。

在经历了有些艰难的第一次集成（第一次集成大多如此）和定期的质量保证循环后，我惊奇地发现，几乎每一个用例都有致命的 bug。在这个为期 15 个月的计划中我们已经过了差不多 5 个月，但是项目工作却远远没有完成三分之一。



我仍然相信，所有的团队成员都会齐心协力按时完成任务。在我们应该通过验收的前一个月，每个人都向我汇报说他们的工作已经完成了95%。但是，当我让一个真实用户来测试这个系统时，她以毫不含糊的口吻对我说："太多地方都有问题，我可不能忍受用这样的产品工作。"这可不像是完成了项目工作的95%。

一位经验丰富的项目经理——帕特里克（Patrick）被请进这个项目，才使一切化险为夷。帕特里克这位项目的救星（今天仍是我的良师益友）把事情都扳回到正确的轨道后，他对我解释了状态的假象。"完成"是由客户定义的，而不是状态报告来定义。

数据库团队报告说完成了95%，但是这与用户是否能使用我们研发的产品毫无关系。即使这个状态报告看上去再完美，它们所展现的项目进展也不是真实的。简单来说，从第一天起，这个项目就注定要失败，因为我没有明确项目的目标。

我最终明白了为什么我们总是需要与用户一起工作，每当实现一个功能就让他们来评估，确保这个功能具有客户认可的价值。这样的话，项目状态报告就转变成了已获价值报告，就能表明实现的已获价值的真实百分比，而不是只显示还剩下多少工作未做。

# 82. 他们到底想听什么



**玛莎·勒加雷（Martha Legare）**
MBA，PMP
美国乔治亚州亚特兰大市

项目沟通有很多种形式，有“四处走动式管理”，也有正式演示。我认为沟通是项目中最重要的活动。

当一个人要向另一个人传达有关软件项目的信息时，最困难也是最常见的方式就是正式演示。一些调查发现，公众演讲比死亡或牙医还令人感到恐怖！

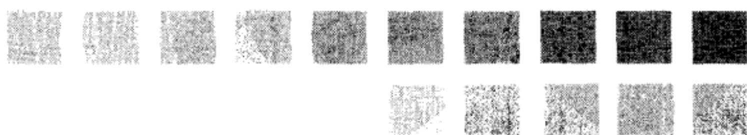
大多数的演示都太冗长乏味，并且充满了琐碎内容。看看你自己最近做的一个 PPT，是不是可以被形象地说成“让人窒息的幻灯片”。

如果你回答“是的”，那就应该重新设计一下今后的演示，使它能帮你与听众进行真正的沟通。问问你自己：“对于我想达到的目标，最好的演示模式是什么样的？”如果你的团队不大，而且你希望能进行互动的讨论，那么有种实用的方法就是使用活动挂图或白板来记录大家关心或达成一致的内容。

然而，如果你的目的是希望主管们能批准一个特别的项目或者同意一种新的项目策略，多媒体幻灯片展示可能会起作用。不过你必须意识到，不管使用什么技术，都是你而不是幻灯片、海报或者激光表演在说服听众接受你的想法。

为了推销出自己的想法，我们必须同时运用左脑的逻辑和右脑的创造力。要使用统计数据作为证据，但是这些数据一定要以一种令人瞩目的格式展示出来。使用易于理解的彩色图表，总结的要点也不能太多。

口头解释每一个要点所隐含的内容，而不是先写出长篇大论，然后待与会人员自己看过后你再大声读出来。



预先使用白板和便利贴来构思你的演示文稿，而不是一上来就直接写幻灯片。便利贴上的头脑风暴能够让你视野更开阔，并且当你要重新表述时，也不会感到数小时的工作都白费了。

将便利贴进行组合，即合并类似的观点，然后思考如何用一种既有意义又令人难忘的方式来组织这些想法。经常要思考两个问题，即：我的中心思想是什么？为什么它对这些听众来说很重要？

通过制造悬念或展示意想不到的事情来抓住听众的兴趣，然后用听众能够设想到的东西来进行详细说明，以支持你的数据。例如，《华尔街日报》曾有一篇文章是这样描述一个给公司财务带来巨大损失的主管：如果你拿着崭新的百元美钞，并将它们一张张摞起来，损失的这些钱的高度都能达到他位于麦迪逊大道办公楼的第 92 层。这是一个令人难忘的形象比喻。

如果拿不准，那就除了保留关键点外，其他一概删掉。如果人们需要了解详细的信息，你可以准备书面材料供大家会后阅读，而且这些可以带走的文件材料还能保证你说的事实不会被误解。这种方式可以保证你能简洁地陈述关键内容。假使你发现会议主持人要挤出时间打高尔夫球，把你的陈述从 30 分钟缩短为 5 分钟时，你也不会措手不及，而是从容不迫，立马就能进行精彩总结。

# 83. 团队士气金不换



**戴维·博克（David Bock）**
美国弗吉尼亚州雷斯顿市

你明白士气正是你需要的，但是它却很难被培养和衡量。一个士气高昂的团队即使偶尔要付出额外的努力，也会高高兴兴，但是一个士气低沉的团队就不会这样。高昂的士气不只意味着工作环境更好，还意味着团队效率更高。

几年前，我与一个有着高昂士气的团队合作。有时候，办公室给人的感觉更像是朋友社区，而不是一个工作场所。工作效率很高。我们偶尔会碰到需要额外加把劲儿的紧急时刻，这时团队成员们会自愿地付出额外的努力。

几年以后，我在一次办公室的冰激凌聚会上见到了这个团队，这时他们看起来并不快乐。团队成员没有相互联谊，彼此的肢体语言显得很冷漠。当我走近时，他们正在抱怨冰激凌上"只有巧克力棒，而没有彩色糖粒"。想想这种变化：同样的团队、同样的项目，但是士气却差别很大。这支团队曾经因为需要而积极加班工作，现在却在围着免费的冰激凌挑三拣四。

这是怎么一回事？原来这个团队来了一个新经理，他做出了许多错误决策。他的失误致使团队走上错误的道路，造成要干的活越来越多。管理层责怪整个团队，而这个经理却不负任何责任，团队从而丧失了信心。项目就变得"工作越多，乐趣越少"，士气也备受打击。

这名经理试图去改善团队的境况时，却使事情变得更糟。他记得，以前团队士气高昂时，大家会偶尔一起出去看电影。因此，他就组织了一次"电影之夜"。但是团队成员已经没有兴趣参与社交活动了，去的人很少。这名经理就开始对大家的表现评头品足，说"某人不参加集体活动"。这进一步打击了士气。



这个经理其实是把因果关系给颠倒了。团队并不是因为积极社交而士气高昂，而是因为士气高昂才愿意进行社交活动。

作为一名咨询顾问，我曾经试图找到种种衡量士气的方法。我曾开玩笑地提出了一些衡量标准，如“停车场在下午 5:05 和 4:55 的停车比率”和“办公室每平方英尺里可以看到的呆伯特卡通人物形象的数量”。但是我意识到，士气不需要衡量，因为它本身就是一个衡量指标。我们要衡量的是团队的态度。士气是一种衡量标准，它衡量的是团队对领导的信任程度、队友彼此间的信任程度以及大家对自己能力的信心。

作为软件项目经理，创建一种士气高昂的工作环境是你的责任。如果团队成员把你当作他们的领导者去尊敬，感觉到能和你开诚布公地交流并且能够影响事情的结果，那么士气就会提高。

士气高昂，员工的满意度就会高，人员流失率就会降低，生产率也会提高。最重要的是，同一群快乐的人在一起工作更会令人感到愉快。你说是吗？



# 84. 让利益相关者全程参与



**卢克曼·拉瓦尔（Lukeman Lawal）**
M.ENG，MNSE，R.ENGR.
尼日利亚拉各斯市莱基

与那些对项目结果有重要影响的利益相关者形成一种良好的工作关系是十分必要的。利益相关者是个人也可能是组织，在组织内部也可能在外部，他们是那些能够影响项目成功的人，并且（或者）也是能被项目影响的人。

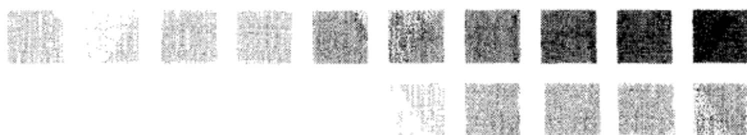
项目经理们应该有一个让利益相关者参与的计划，要确定项目利益相关者的名录，对他们影响力作评估，对他们对项目的支持力度作评估。用这个计划有意识地团结一切支持力量。

尽早让利益相关者参与进来，并让他们参与到底。一定要知道他们支持项目的商业需求。使工作方向与所有关键利益相关者而不仅是几个高层利益相关者的需求相一致。

有一点总是必需的，要去发现关键利益相关者需要哪些条件才能成为项目的鼎力支持者，以及这个项目怎么做才能在他们和你的团队之间形成共赢。

一旦项目计划反映出了关键利益相关者的需求，就坚持要求他们承诺提供所需资源来支持项目。这就给了你一支投资者部队，当那些既不出钱又不出力的外围利益相关者试图阻止这个项目时，就可以用他们抗击这些人。

在项目团队中指派唯一一个责任点来协调管理利益相关者，同关键利益相关者主动保持良好往来，让他们适度参与项目。创建利益相关者交流计划，控制好交流的频率、交流的内容和交流传递的方式。与每一个利益相关者的接触方式和频率可能都会有所不同。



安排同小部分的利益相关者的联席会议，减少部门之间的分歧。团队里负责利益相关者事宜的责任人要继续跟进这项工作，确保关键部门或小组都能作为一种资源，在核心项目团队或是扩大的项目团队中得以体现。

在每一次会议上，与决策者一起检查利益相关者的成功标准，使其保持一致。

还有一些其他的建议。

- 即使你不同意，也要尊重利益相关者的商业需求。
- 确保不泄露敏感问题，建立互信。
- 创建利益相关者联盟以支持你的计划。
- 让利益相关者积极地、有意义地参与到项目中来。
- 收集利益相关者的想法，使用他们的建议。
- 保证利益相关者被及时告知。
- 利用那些有积极态度的人，如果可能的话，给予他们实实在在的授权。

但是，要对未来的挫折和障碍有所准备，一旦问题出现，随时解决。前进的道路上总是会遇到坎坷，所以不要让自己或利益相关者背上追求完美的包袱。当问题出现时，就去解决它们，并且从中吸取经验教训以防在同一个地方再次跌倒。

尽早让利益相关者参与进来，并让他们一直作为项目伙伴，这样你就可以防范叫停项目的人。利益相关者有可能不了解项目管理的细节，但是如果你帮助他们实现了商业目标，他们就会与你一起庆祝项目的完美表现。

# 85. 计划的价值



**德里·齐美尔（Derry Simmel）**
PMP，MBA，FLMI
美国南卡罗来纳州切宾市

“在战斗准备过程中，我总是发现计划是没有作用的，但是计划过程却是不可或缺的。”

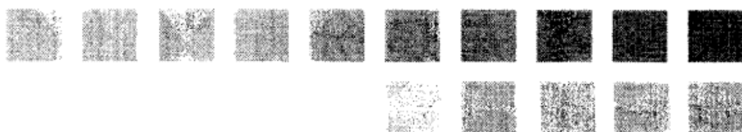
——德怀特·艾森豪威尔，第34任美国总统

一些项目经理喜欢引用巴顿将军说过的一句话：“一旦与敌人接触上，就没有计划可以幸免于难。”他们推断说，制作计划就是浪费时间，因为计划从一开始就无效。这样的态度注定了许多项目会失败。总是有许多项目经理声称行动比计划更重要，行动才有魅力，而做计划则很无聊。

计划可能会无聊，但是制定计划的唯一目的并不是编制一堆文件。艾森豪威尔就意识到，计划的目的促使你去思考项目。计划的过程会加深对项目的理解，你要处理任务、预算、资源、风险、时间表，等等事项。做计划时，你就能深入了解成功所需的是什么。你的计划同样会帮助你明白完成目标所需要的条件和方法。完整的计划是价值连城的项目沟通方法。

计划文件记录了讨论的内容和所做的决定。文件的存在并不是为了让行动一成不变。不幸的是，最原始的计划会很快失去它的价值和实用性。这就是为什么我们会有两类计划：初始计划和进行中的计划。

初始计划的目的在于为项目设定方针路线。初始计划从全局的观点去看待整个项目，并考虑所有的因素（风险、时间和质量等）。初始计划设定项目的意图，并绘制一个实现目标的合理路线图。随着获取的信息越来越多以及形势不断改变，路线图也会随之改变，这是很正常的。但是改变目标却是不常见的，想这样做时千万要谨慎。



为了编制初始计划，你必须思考你的项目，明白风险所在和限制条件，并找到一个通向成功的道路。作为一名项目经理，你要和团队通过一系列的沟通来制定计划。创建一个达成共识的初始水平是至关重要的，特别是在面临“第一次接触”的敌情时。

第一次接触会让计划有所回报。由于每个人都明白这个计划，所以大家都会根据项目意图自主地做出反应。了解了所有的目标和限制条件，团队会迅速地作出正确决策。这时就需要制定进行中的计划了。

进行中的计划是在现有计划基础上开始制定的，并且根据新出现的情况不断进行修订。你计划在五月一号开始研发过程，但是现在看来六月一号才能开始。你如何弥补这段时间？你能够把资源分配到其他能够在五月完成的项目上吗？你需要更多的时间和资金吗？这是你做进行中的计划时需要考虑的。你不断计划、调整，然后执行。

每次做计划的时候，你就会思考，并且就你的项目与别人进行交流。这些必要的和基本的活动一定能保证产出远远大于投入的成本。



# 86. 不要总是扮演“信使”



马特·萨克斯科（Matt Secoske）
美国内布拉斯加州奥马哈市

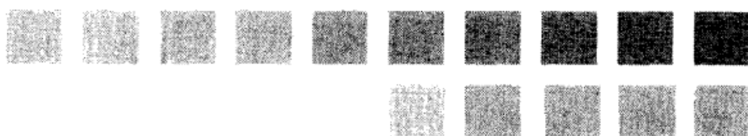
软件项目经理的一项最重要的任务就是促成团队中各成员间的直接对话。遗憾的是，在我曾经历的许多项目中却出现了截然相反的情况。项目经理成为了瓶颈，所有的交流都得通过这个瓶颈。这个项目经理俨然是一个信使，负责将那些宝贵的点滴信息从一个成员传到另一个成员那里。

项目会不断发展成为一个相互交织的有机整体，因此，信息就变成了代码库赖以生存的空气和水，直至实现项目的最终任务。所有的团队成员都依靠信息的不断交流。但是，如果要求相关人员必须通过项目经理来沟通所有信息，那一定会出现一些难以克服的问题。

在获得最近的更新信息后，项目经理或许没能准确地判断出哪些研发人员需要了解这些信息。而发出消息的人则认为，只要把信息传递给项目经理就算完成任务了。一旦发现信息传递通道上出现了遗漏，第一个发出信息的成员可能都记不清他先前传递的是什么信息，因为他已经转移到新的问题上了。项目经理会被那些他可能并不理解的技术报告所淹没，很快就不能再胜任项目智慧的唯一信使这一岗位了。

还有一种角色，他的破坏力比“信使”更强，即一个善意但一窍不通的项目经理会变成“扰频器”传递错误信息。随着项目的成长，所需要的非技术信息量也在增长，以确保项目顺利实施。

开发人员需要明白业务规则，业务能手需要知道交付物的进展状况，而各色人等需要理解项目的进度、成本和质量水平。



随着信息量的增长，出现下面这种情况的可能性也越来越大：一个绝非全能的项目经理误传了信息。扰频器敲响了！例如，一个表面上看似对项目无关紧要的业务规则，等大家搞明白其真实意图，可能真的就成了一道主要障碍。为了弥补损失，需要对代码库做出相当大的改变。

项目经理需要在合适的时间召集到一群合适的利益相关者来讨论合适的话题。找个时间让来自不同部门的人员聚在一起，这件事似乎会让人望而生畏。通常会怎么做呢？为了解决一个进度安排的问题，项目经理会说："谢利，我直接去找鲍伯，有了结果再来找你。"对于那些简单的、非技术性的问题，这么做可以。但是，习惯这样做事会逐渐使你变成一个信使或者扰频器。转述过程中信息丢失是不可避免的，最终又要浪费大量时间来处理善后事宜。

只要有清晰、开放的信息交流通道，并且对所有讨论和决策进行存档，那么，团队成员之间就可以直接进行交流互动。这样一来，就不会出现信使和扰频器项目经理了，软件项目也能不断前进。

# 87. 有效管理交付产品



**埃尔纳尼·马奎斯·达席尔瓦（Ernani Marques da Silva）**
MBA，PMP，PgMP
巴西圣保罗州迈里波朗市

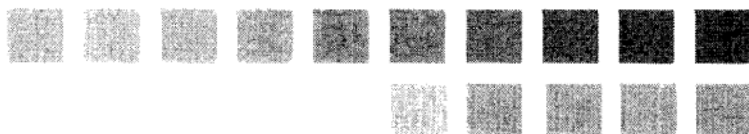
项目都是由一系列交付产品所组成的，当项目完成时，这些交付产品再组成一整套完整的产品、服务或结果。对于软件研发项目，要想最终结果令人满意，对所有这些组成部分的整合是至关重要的。当然，你开发的软件不同，这些组成部分也会不一样。因此，交付产品是最主要的部分，我们应该根据以下几个小窍门来积极地计划、控制、监督和管理这些部分。

- **明确交付产品。**明确了交付产品，等于就有了完整解决方案的轮廓，明确创建、交付产品的顺序，明确监督、控制这些产品研发和交付的标准，并根据计划的基线和明确的标准对交付产品的进程进行积极的监督。

  将交付产品分解为局部代码包是非常重要的，每个开发出的代码包都提供一项专门的软件功能。对于复杂的项目（环境）以及由第三方开发的项目，这一点尤其重要。不要等到最后再接受一个完整的工作包。有一个好办法就是，安排项目的各部分一个一个地交付，再根据事先制定的流程将它们布署给软件团队的其他成员，以便他们在各自的开发工作中使用。

- **监督和控制交付产品。**一旦明确了创建、监督和控制工作包（功能代码）的方式，就必须对创建阶段进行积极的监督和控制，以核查工作是否按计划进行。检查点、度量标准和关键绩效指标（KPI）应该告知项目团队的所有成员。

  在检查点，应该根据基线和趋势分析对关键绩效指标和度量标准进行评估，以明确差异。这样，就能根据实际标准而不是直觉或传闻来采取纠正措施。



- **管理交付产品。**一旦交付了预想的作品，就应该将程序交给一小部分用户进行测试和布署，这样可以在最终完成之前来核实工作是否符合要求。这种方法有助于识别问题，这样，在整个用户组全面布署这套软件之前，还可以采取必要的纠正措施亡羊补牢。

在运用这种方法的过程中，有一点很重要，一定要记住，那就是所有的微型交付产品要按阶段准备好，然后再以一种整合的方式对这些产品进行测试。如果你一直等到完整的代码集都提交后再进行测试，那么，你收到的这一堆代码中可能会有许多未知的错误（缺陷）和意想不到的行为。因为产品、服务或结果是在这些问题未解决时创建的，所以这些问题的影响会未被肃清，会被放大。到这个时候再对所有这些不靠谱的程序进行修复，成本会很高，用时也会很多。

你可以权衡一下这样开发所必需的复杂度和厂商对你的公司环境和系统的体验，看看这种方法是否适合你。一般而言，对于那些比较复杂的解决方案，或者那些新技术和新方案，这种方法是最有效的。



# 88. 我们只是项目经理，不是超级英雄



安吉妮·肖克-史密斯（Angyne J. Schock-Smith）
PMP
美国新泽西州菲利普斯堡

如果你是IT业的软件项目经理，这条建议一定会帮到你。就算你不是IT业的，不论你管理的是什么类型的项目，这条建议差不多都同样适用。

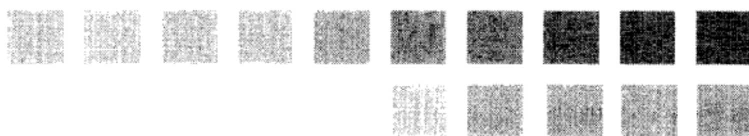
我给人上项目管理技巧的培训课时，例行的开场白是："我们只是项目经理，不是超级英雄。"当讨论一个好的项目经理的性格时，我喜欢的一句话是："伙计们，拿出你们的斗篷！只有超级英雄才能完成软件项目经理必需完成的所有工作，而且只有超级英雄才能做好。"不过，因为培训才刚开始，我必须给大家一些希望。因此我说："好的，我们中的一些凡人，照样能够成为优秀的项目经理。有什么窍门？"

我认为窍门分三个方面。

- 弄清你个人的优势和不足。
- 弄清你团队中成员的优势和不足。
- 基于以上认识与团队成员建立互补伙伴关系，取长补短。

怎么了解自身的优势和不足？

- 找出自己过去参加过的所有性格测试或工作行为测试的结果。
- 基于这些过去的评估结果进行推断时要诚实。当时参加问卷调查时，自己诚实吗？那个"标签"仍然适合你吗？或者从那个时候到现在，你又变得成熟了，想法也不一样了？那个标签里提到的哪些优势和不足能够最准确地描述现在的你？



- 别想着哪个标签会成就最优秀的项目经理。这个问题没有标准答案。优秀的项目经理必须能随机应变，能够区分具体情况，跳出自己的思维定式，以最有效的方式对具体情况作出回应。
- 根据手头上的信息，列出自己当前的优势和不足。把它放在一个你随时能找到的地方，当对自己有更深的了解时，立即更新这份清单。

上面这项工作做完之后，剩下的事情就容易多了！用一份现有的优势清单来评估你的团队。然后，看看团队中哪些人的优势正好是你的不足。

对于我来说（我是一个表现型的人，如果你了解社交风格分类的话），我的弱点在于不太注重细节。我总是需要某个人（一个分析型的人）来督促我，让我关注细节！如果你要试图超越自己的能力来取悦他人，那可能就需要有人用更加强势的手段来帮你推动项目，而不是随着自己的性子来做。

确保和你共事的团队成员中有人跟你能力互补，正好弥补你的不足。但是你也没必要告诉他们这一点，对吧？保持一点神秘感，说不定你能让整个团队相信你就是一个超级英雄。反正，我是不会对任何人说的。

# 89. 增加交流：时常召开即时会议



**理查德·谢里登（Richard Sheridan）**
美国密歇根州安阿伯市

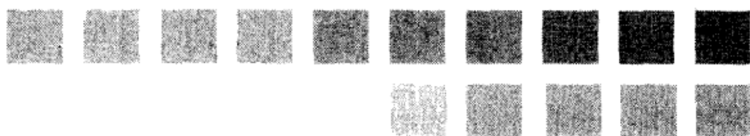
软件项目经理经常会落入可怕的陷阱，即为团队安排定期的会议，而这些令人痛苦的会议往往会带来一个遗憾的、非计划中的结果，那就是实际上减少了交流。一个有史以来最可怕的会议就是经典的周一上午通气会。难道周一的境况还不够惨吗！

如果你不相信大多数会议都应该取消，可以做个试验。作为软件项目经理，你不要露面。问问你比较信任的一个同事，看看你不在的时候他们开会了吗？

如果会议只是在老板或者项目经理在的时候才召开，那么就取消它。你的团队告诉你，他们并没有从中获得价值。千万不要只因为一个人感到有价值而开会。

在我的组织里，我们竭尽所能取消那些徒劳的会议，代之以团队成员之间进行更简单的沟通。例如，我们让团队整天、每天都呆在一个开放的大房间里，没有墙，没有办公室，没有隔间，也没有门。这样，当我需要从某个人那里得到一个答案时，我可以简单地说："嘿，詹姆斯。"在 30 秒之内，詹姆斯和我就对必要的信息进行了交流，并且在没有实际移动（也没有来回发邮件）的情况下又回到工作上。

设想开一个有 60 人参加的公司全体会议，这个会议开起来可以非常简单。一个人开头喊："嘿，大家注意啦！"每个人停下手头的工作，回应道："嘿，里奇！"几分钟后，会议结束，每个人都没有从他们的座位上起来，很快又回到了原来的工作中。



我们的例会包括每周“展示和陈述”，为项目投资方讲解项目进展；每周“策略规划”，来确定项目范围；每天站立会议；以及每周启动会议，以头脑风暴方式讨论怎样齐心协力完成下一周的项目目标。这样的会议有一个共同的结构，让每个人都可以轻松参加，并感到愉快。

试着在一周之内每天都召开一次站立会议，看看是否受欢迎。以下是我们的一些经验，可以帮助提高会议的效率。

- 邀请所有与项目有关的人。我们通常有五六十人参加这样的会议。
- 用一个能让每个人都听得见的闹钟来召集大家开会。一个公正无私的装置更容易把大家召来开会。我们用的是一个带闹钟的飞镖盘。
- 找一个发言凭据。我们用一个塑料的北欧海盗头盔来控制会议。站（不许坐）成一圈的人来回传递这个头盔。拿到头盔的人可以发言。
- 让大家汇报近期完成的工作、正在做的事情，以及他们需要帮助的地方。会议当时不提供帮助，是在会后。

我们的站立会议一般只召开 13 分钟！宣布，集合，进行，给每个人发言的机会，结束，回去工作，这一切都在 13 分钟内搞定。我敢说，大多数公司都不可能在 13 分钟内完成一个有 60 人参加的实用会议。

# 90. 用灵活性简化项目管理



**克利斯那・卡达利（Krishna Kadali）**
工科硕士
印度海得拉巴市康达巴

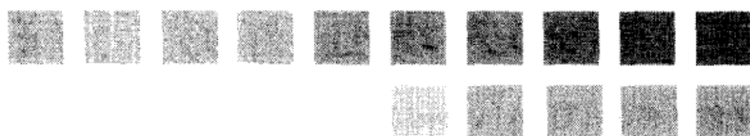
17 世纪的著名武士宫本武藏（Miyamoto Musash）坚信这样一条原则："不要对任何一件武器或者一种作战方式产生依赖。"这条忠告对于我们管理项目来说也同样适用。我们绝对不能过于依赖某一种管理原则、软件工具或编程语言，不能贪恋唯一的武器。只有灵活地看待资源，审时度势进行不同的配置组合，才能随时待命，以最佳的方式解决客户的问题，拿到项目。

定义项目需求时，从一开始就要有一个开放的思路。如果在和客户讨论之前就已经选定了自己的武器和作战计划，你如何能保证自己的解决方法就是最好的呢？

在选择武器之前，先要摸清客户的所有需求。这个客户打算用你的新软件来解决什么问题？这类问题通常会反映出某种商业逻辑，即当前的资源并没有足够丰富的特性或功能。其次，考虑公司中现有的要素和系统。熟练而恰当地重用现有系统，能够缩短并减化项目。

可以利用的企业要素非常多，包括由现有设备构成的公司基础设施、器材和软件、商业和个人数据库、编程工具以及人力资源技能。

对客户需求进行以资源为导向的分析，从而使开发的软件或其他可交付的项目产品满足这些需求，这样做会揭示出一些在企业内已经存在的潜在资源："我们的用户必须要同时看到 XPP34 呼叫中心系统和我们的公司应收账款系统。你们的产品需要合并或者兼容我们原来的 4465IL 软件。"



你的任务就是问对问题，了解客户所期待的各种最终执行结果，以及现有资源（软件或系统）和你要开发的软件有什么内在关系。你的客户在未来可能需要对产品进行一些改动，而你或许可以依靠这种面向资源的方法来为这些改变打下基础。经过这么一分析，你的软件摇身一变，也将成为一套崭新的资源，它不仅能够解决今天的问题，而且还能用到未来的项目上并与未来的软件互动。

只有对消费者的需求和公司拥有的资源了然于胸，才能选择最好的武器来解决问题。实际的开发首先要关注几个关键的需求，因为提交这些需求可以帮你赢得客户的信任。为实现这些关键要求，需要一些特别的模块和服务，应该首先开发这些模块和服务。

面对千变万化的需求，要想处理好软件项目，就得让你的大脑以开放的思维来不断创新软件设计。这种灵活性将简化项目管理，而且还能不断创造新武器和新计划，让你每天的工作充满乐趣又丰富多彩。



# 91. Web为现在指明了道路



**戴维·伍德（David Wood）**
美国弗吉尼亚州弗德里克斯堡市

有一个慈善组织的口号是“我们站在前人的肩膀上，后人站在我们的肩膀上”。这句话同样也与我们软件开发者密切相关。每当一种新架构、一门新语言或者一个新平台崭露头角时，我们都免不了感叹：“这就是解决我们所有程序设计问题的答案啊！”但是，它可能解决了今天的问题，明天我们又会面临新的挑战。

目前，我们确切知道的，是有一个软件架构的使用人数达到了数十亿，与此同时还能保持其健壮性[①]，个别组件的损坏并不会影响它的运行，它就是万维网。Web 是最大、使用最广泛、迄今为止人类所建造的拥有最健壮的信息检索功能的系统。

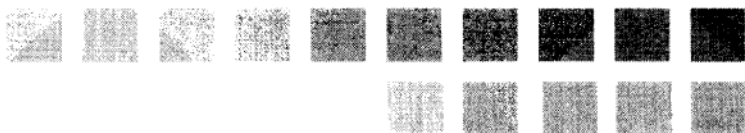
为什么 Web 能运行得这么好呢？罗伊·菲尔丁（Roy Fielding），赫赫有名的 Apache 项目[②]的创始人之一，曾经研究过这个问题。菲尔丁评估过早期 Web 的一个理想化的版本，并且从中得到了架构风格的元素。

结果，就是一种新的软件架构风格[③]，与人们喜爱的 Web 具有相同的特点。这种架构风格非常健壮，不会因个别组件的变更和损坏而受到影响。这种架构风格能够分离相关问题，让我们不必担心具体实现的细节（例如使用什么编程语言）。这种架构风格使用一种通用语言（一种在非共同母语人群中使用的语言）来交换语言中立的信息请求。这种架构风格可扩展性极强，而且无状态。

① 健壮性：面对各种变化（有时候是不可预知的变化），能够确保功能破坏、变更和损失最小的能力。

② Apache项目：是一个开源项目，旨在为现代操作系统开发和维护自由的Web服务器软件。

③ 参见Fielding的论文：*Architectural Styles andthe Design of Network-based Software Architectures*，网址为http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm。——编者注



并不是每个站点都遵从这些指导性原则，但是，大部分还是遵从的；并且Web从整体上也是遵从的。然而，这些只是创新性架构进一步发展的基础。

我们能从Web的成功学到很多重要的东西。或许，最重要的就是摩尔定律[1]，现在这个定律使我们能够实现大量抽象的系统设计概念。我们不是用它讨论软硬件的极端高效发展，而是可以考虑其极度稳定、极度健壮、极度扩展性和极度灵活性。我们同样承认现在的架构还不够好，知道它们仅仅是未来创新的基础。

像Web这样的分布式系统是很难设计的。或许，这是因为我们每个人都是一个独立的个体。我们看待软件系统就像个人看待世界，好像它们都是由一个个体创建的、集中式的一样。尽管如此，Web分布式的系统已经给我们指明了道路。分布式系统很难概念化，并且更难设计，但是确实值得为创建它们而努力。

自然地，技术在不断改变。同样，观念和方法也在改变。菲尔丁所描述的简单Web的概念并不是现代的Web，更不是将来的Web。Web也不是总可以指出方向。适应新系统的关键应该是现在就把灵活性设计到我们的系统中来。只有这样，我们才能设计出一个有活力、有生命、适用性强的软件系统，而且这一系统随时可以融入新的发明，可以为后人提供肩膀。

① 摩尔定律：指的是从长期来看，计算机硬件中可以包含在集成电路中的晶体管数量会按照几何级数递增，而成本上升不多。

# 92. 开发者厌烦状态报告，经理们却喜欢



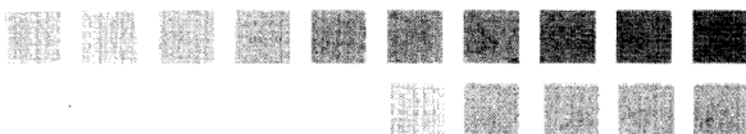
**帕维尔·西姆沙（Pavel Simsa）**
PMP
美国华盛顿州贝尔维尤市

根据在世界上最大的软件公司工作的经历，我能证明开发者确实讨厌状态报告。为了要完成状态报告，他们每周要花数小时去记录那些对他们而言显而易见、冗繁多余的信息。

然而，对于身为软件项目经理的你来说，要利用这些数据了解项目进展的情况，并且还要向上级管理层汇报。一般来讲，一个项目经理会同时指挥五到七个项目。不论是从你自己还是从你的高层管理团队来讲，你都需要收集并传递这些项目数据。

这里有几个小诀窍能使你的开发者减少对状态报告的抵触，不论你要求递交报告的频率是多少。

- 尽力向他们说明这个报告为什么对团队里的其他成员或者其他部门很重要，因为其他人需要根据团队的工作进展制定计划。认识到能帮助同事，人们的工作会更努力。
- 如果项目进展缓慢，应该了解团队正在做什么。他们是不是在学习新的工具或者语言？是否这周出现了意想不到的问题或者挑战？当你编写状态报告时，要增加一些解释性的信息来帮助其他人了解这些数字。
- 给予适当的认可。如果你知道项目面临哪些问题和挑战，就不会让有价值的成果埋没在进度报告中。对于那些做出了有益且计划外贡献的人，送他们一张附近咖啡店的拿铁咖啡优惠券。并给整个部门发一封感谢邮件，就写“干得好，谢谢你，（名字）”。让所有人都认识到这项工作对项目全局的重要作用。



- 如果你管理的开发者不止一个，那就对团队采取一些激励措施。“如果我在一个月之内每周五下午 3 点之前都能收到你们所有人的状态报告，那么再下周五的下午放假”或者“我给大家带午餐”。没有人希望自己成为拖团队后腿的人。
- 让写报告成为一件容易的事。给大家一份提交状态报告的模板或者电子工具。准备好重新组织报告里的语言，让每个人都能理解这些内容。你们副总很有可能不明白“lcl check-in to main build lab”，你可以把它改成“实现了功能里程碑 2；一切按计划进行”。

关键在于，你要从他人的角度来看待这项完成周期性状态报告的任务。状态报告很重要，因为每个人都需要知道项目的进展情况。高层管理人员关心里程碑，而商务管理人员关心预算。作为项目经理，你的工作就是确保每一个利益相关者都知道项目的实际进展情况，但也要知道，如果没有你的帮助，有一些利益相关者可能就理解不了报告里的技术细节。

找一种有效的输入工具，尽量透彻理解报告背后的任务。你需要创建一份全面的状态报告以满足所有利益相关者的需求。



# 93. 你没有控制住



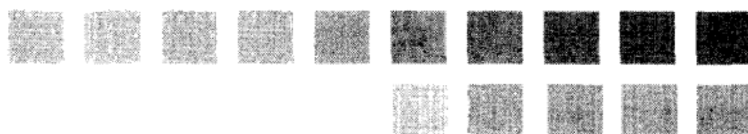
**帕特里克·夸（Patrick Kua）**
英国伦敦

我依然记得曾指导过的一个项目团队。那位项目经理很明显想要控制项目的一切。他像是有强迫症似的，希望参与讨论所有“关键的”决策。他积极地主持每天的站立会议，并且由他一个人决定在项目回顾例会上谁来发言。我对这个团队的第一印象是他们很有组织性，但很有意思的是，我注意到项目经理在场和不在场的讨论质量有天壤之别。

当我与团队中的几位成员单独谈话时，他们承认自己很讨厌项目经理主持召开的所有会议，他们只希望会议快点结束。他们觉得这种会议是在浪费他们的时间，因为他们的真实想法并没有得到重视。他们列举了几次同项目经理交流的情况，他们说的都是项目经理希望听到的，这样才能把问题说完。当他们遇到需要解决的问题时，他们会去找技术领导。因为技术领导更愿意进行开放式的讨论，因此也能更高效地解决问题。

我从这个团队汲取的教训就是，表面上对局势的控制，并不意味着你就真正地掌控了局面。事实上，急于寻求控制有些时候会造成负面影响。一个有经验的、组织良好的团队并不欢迎一个控制欲很强的人，对那种不能带给团队利益的控制更是深恶痛绝。

了解团队动态和不同的领导风格对项目经理很有用。不同的项目和不同的团队会需要不同程度的控制。组织良好、表现优秀的团队经常会反感过分的控制，除非他们发现这种控制对他们有帮助。



控制经常被看作“管闲事”，哪怕团队成员口头上同意了，他们会后的表现可能并不会如你所愿。但是，对于一个新成立的团队来说，多一点控制可能会为团队指明方向，并能建立更加清晰的项目目标。

优秀项目经理所实施的控制一定是恰如其分的，不会妨碍团队成员给项目带来技术、经验和关系。他们能识别出在什么时候增加控制可能帮助团队实现最终目标，同样也能识别出在什么时候同样的控制会减缓项目进度。

没有什么比让非IT项目经理去领导软件开发项目更难了。由于憎恶外部对自己工作的干涉，团队常常会使项目经理带队的经验大打折扣。

但是，组织管理技巧、让项目与公司目标保持一致的能力，以及协调高级管理层与客户之间冲突的水平确实可以让团队成员少做很多无用功。



# 94. 分享观点

**贾里德·理查森（Jared Richardson）**
美国北卡罗来纳州莫里斯维尔市

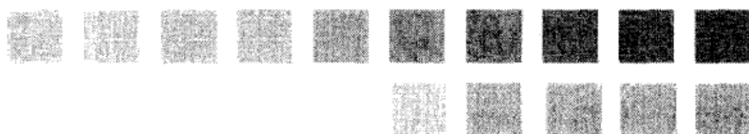


你是否在和一群白痴一起工作？你的团队成员是不是想让这个公司倒闭？有时的确会有这种感觉，但通常不会有这种情况。其实，每个人都渴望成功并且对自己的贡献感到自豪，无论表面看上去他们多么像是在破坏项目。他们在做自认为是正确的事，但是每个人都对“正确”的事有自己不同的想法。

作为软件项目经理，如何让大家一起协调工作呢？要知道多数成员是在迷茫中工作的。他们不知道项目为什么重要，它在公司更大的战略中的分量，或者为什么截止日期是 6 月 17 日。因为他们不明白相关决策是如何制定的，所以这些决策看起来就显得有些武断和不合理。每个人都在挣扎，努力在没有任何明确信息的帮助下对模糊的形势形成一个清晰的认识。这样的话，如果大家对项目最终目标的看法不一致，我们还会感到惊讶吗？

想要驱散谜团，你需要与团队成员分享关键信息，使他们明白你的努力。让他们知道项目需要在六月中旬完成，因为只有这样才能打败竞争对手，使自己的产品领先他们三周上市。帮助他们理解这个项目的目标是为了配合一个更大的有关国际扩张的公司战略，或者说你的客户就指望这个项目来提高现有的、老化的生产线的利润。

如果你对信息共享缺少经验，那就要谨慎从事。作为团队的信息通道，你同样要提升团队的士气。当你想要去抱怨其他团队或者经理，或者你团队的某个成员时，你的消极态度会像流感一样快速传遍你的团队。并且，像感染病毒一样，它会降低团队数天的活力。



分享项目信息的一个好办法就是每天开例会。如果是十人左右的团队，仅仅利用一二十分钟就能开一个高效的会议。每个人有一两分钟来告诉大家他工作中的新进展，如果需要，还可以寻求帮助。这些快速的“站立”例会是软件项目经理分享项目最新情况的绝佳方式。

当你每周（或每月）才开一次例会，你可能会忘掉一些重要的信息。毕竟，当整个团队最终聚到一起开会的时候，很多信息都成了旧闻了。或者，由于你延误了与大家分享预警信息的时间，一个本来可以避免的问题却爆发了。如果你仅用一次冗长的会议将 17 条“重要”信息揉在一起，在你分享完毕后，团队成员只会变得呆滞迷茫。

记住，你的团队和你公司里的每一个人都希望成功。不但要分享你自己的观点，每个人都要分享他们自己的观点。你将会发现，很多你原以为是想让公司倒闭的白痴，其实能和你肩并肩共同解决团队面临的挑战。



# 95. 善于支持的组织就能获得成功



**辛西娅·伯格（Cynthia A. Berg）**
博士（准博士），PMP
美国亚利桑那州格伦戴尔市

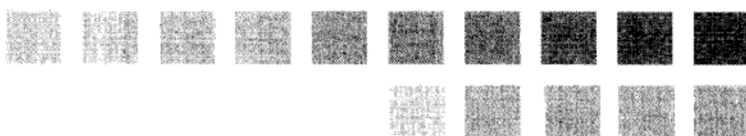
如果公司没有作风险计划、不积极发现问题也不及时解决问题，那么公司的文化可能就有问题。项目团队中那些唱反调的人常常被看作是麻烦制造者。如果公司急于“斥责唱反调的人”，团队成员就不会再叨叨那些让你烦心的事儿了，甚至还会有意隐藏项目中的问题。

这种文化基因鼓励指责行为，但是这种指责行为会损害整个公司、员工个人和客户的利益。软件项目经理的任务就是保质保量地如期交付项目，并且意料之外的事情越少越好。由于没有人提早指出潜在的危险，因此经常会有“使人意外的”突发事件。这些突发事件很少是使人“惊喜”的好事，而是让人惊讶的坏事，比如，由于开发人员隐藏了一些问题，原先的期望和计划不可能实现了。

明智的管理人员会大力支持那些有助于开发人员提高效率的态度和行为。比如，对人力资源政策和激励措施进行评估，以确保能够鼓励开发优秀产品和服务的行为。

一个经典的反面例子：公司官方“鼓吹”团队意识，但却一直奖励个人贡献。大家都很聪明，他们知道怎么做对自己最有利。如果上级管理层能够在宣传和实际行动上保持一致，使之形成一种激励高效行为的工作环境，那么个人和组织都能够茁壮成长。

如果你发现自己处在一个没有支持或沟通不畅的公司里，下面有一些方法可供使用。



- 多方打听，一定要弄明白自己所负责项目的范围，这样才好开展工作。
- 挑一些团队成员和其他相关人员。尽可能让他们参与头脑风暴、制订计划和执行项目。
- 让做具体工作的人都能充分参与到项目变更和决策中来，至少在他们完成与这个项目有关的所有工作之前应该如此。
- 永远做一个诚实的软件项目经理。不要企图掩盖问题或通过简单地大事化小来回避冲突和不愉快的讨论。
- 在你的项目团队中创造这样一种氛围，即你很愿意看到整个公司都来指出你们的错误。

项目经理必须客观地对待项目。他们扮演着并不令人羡慕的角色，他们首先要忠心于支付自己工资的公司，同时又要和开发者建立一种相互信任的关系。如果项目前景并不乐观，精明且有原则的项目经理就应该建议取消这个项目，除非外部问题都得到了解决。

我们都希望自己工作的公司有既定的战略来支持新软件项目研发。但遗憾的是，就算是同一个公司里的不同部门，这种能力也会相差很大。既然营造一个更容易获得支持的氛围能让所有人都受益，那么，软件项目经理的职责也应该包括这一条：提醒上级管理层注意项目优先次序的安排和行为奖励之间的文化冲突。

# 96. 建立项目管理控制



**埃尔纳尼・马奎斯・达席尔瓦（Ernani Marques da Silva）**
MBA，PMP，PgMP
巴西圣保罗州迈里波朗市

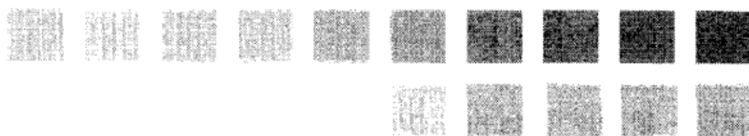
与项目打交道的团队可能很庞大，包括团队成员、卖方团队成员、客户或者项目投资人、运营团队、合同团队、财务团队以及其他利益相关人。在这种情况下，由于项目涉及一大群人，各种不同的状况都可能会威胁到项目。

如果你过去是做技术的，那么跳出你所在的部门，从更高的项目管理层面来看问题，对你当好软件项目经理很有帮助。

控制是一种管理方法，用来制定、沟通、执行以及监督政策、流程、方法和其他用于运行项目的所有方面。建立有效的项目控制结构和流程，能够保证项目的一致性，监督并控制机遇和风险，辅助决策以及按项目计划交付。让你能够恰当地处置风险，而后满足项目需求。

要想有效率，项目控制应该提前计划。在框架内列出相关的事项，包括：控制的目标和宗旨；结构；原则；方法、流程和标准；沟通；报告关系；升级流程（内容、时间、方法以及由谁操作）；工具；定义清晰且适用的责任和义务；衡量和衡量标准；质量；会议和指导委员会；审计；监督和控制。

要记住，很多因素都会影响控制，包括环境、部门、产业、公司文化和法律。例如，作为项目经理，你在一个职能组织里可能直接向职能经理汇报，而不是向投资组合经理或计划经理汇报。在一个项目化（所有的工作都设置为项目）的组织中，项目经理拥有最高的权利。但是，在职能型的组织中，你可能要直接向上级主管或者部门经理汇报，这会大大削弱你的权利。



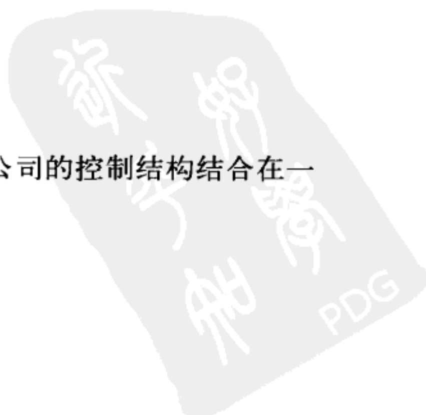
因此，当你计划或者定义控制结构时，就要考虑到管理阶层。如果需要，依据项目发展的情况，这个结构是可以修改的，以保证项目与其计划的目标及宗旨相一致。在一个庞大的项目里，保持你的工作与更大的计划或整组项目目标及宗旨相一致。即使你管理的是一个很小的项目，同样也要创建或调整项目控制模型。

通常，项目管理办公室（PMO）有责任规定和管理与项目相关的流程和手续，并且创建应该遵循的模板。

项目管理委员会要确保实现项目目标。它为恰当地解决项目风险和其他问题提供支持。委员会的其他功能如下：

- 批准项目计划和对计划的变更；
- 为撰写进度报告收集资料；
- 落实政策、程序、标准和需求；
- 在风险和问题方面提供指导；
- 审查项目的进展情况。

当项目涉及其他公司时，应该将项目控制与其他公司的控制结构结合在一起。



# 97. 我讨厌你的网站的 9.7 个原因



**芭比·戴维斯（Barbee Davis）**
文学硕士，PHR，PMP
美国内布拉斯加州奥马哈市

大多数公司不知道软件开发不同于网站开发，因此，他们会叫软件项目经理和软件开发人员去做网站设计。这里有 9.7 个问题，如果你的网站存在其中任何一个问题，我都可能因为生气而不会和你的公司做生意。

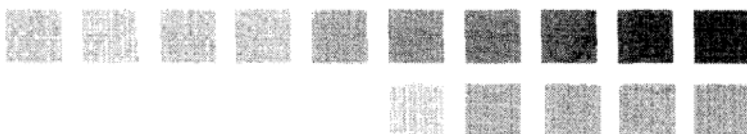
(1) **一上来就是一个慢吞吞的 Flash 界面**。不让我跳过 Flash 动画，随后访问每个页面都会让我无穷无尽地等下去。我的第一反应就是你的竞争对手在呼唤我。

(2) **猝不及防、难听刺耳的视频剪辑吓我一跳**。我可能在上班，可能身边有熟睡的孩子或配偶，也可能在偷偷地买礼物，想给谁一个惊喜。如果你真的想让我离开，那就不要让我看见关闭按钮。

(3) **禁用后退按钮**。你觉得自己很聪明，让我不能返回找到你网站的搜索引擎，但是我不会同一个地方跌倒两次。下一次，我保证不会再点击你的网页，或者购买你所销售的产品和服务。

(4) **选择一个让人看不清的配色方案**。在暗灰色的背景上使用浅灰色的字体，看上去很特别，但是这样可读性并不好。我只要点一下鼠标，就可以看到另外 25 个网站，看这些网站我都不必担心把眼睛弄瞎。同时，反白字（黑底白字）会使我剪切和粘贴变得很困难，这意味着我不能保存使我决定购买产品的信息。

(5) **忽视我的便携式设备**。我可能会随身携带 iPhone、Kindle 或者上网本。如果你没有一个快速、低耗电的移动版界面，就不是一家我要找的现代公司。我们正在开历史的倒车，客户端 / 服务器模型的感觉荡然无存，让人依稀觉得手里的“哑终端”连接到了远在天边的大型机上。你得好好考虑一下了。



(6) **没留下电话，让我联系不到人工服务。**如果我遇到问题，即使我已经用尽了它的有限帮助，仍然坚持让我只使用你的网络支持功能，是绝对错误的。我会选择你的竞争对手。难道你不想知道自己的网站什么时候崩溃吗？

(7) **非要让我打电话了解情况。**除非你在出售前所未见的新产品，比如说土星传送器，否则有什么可保密的？直接展示出来不就完了。小心竞争对手打电话套取你的信息。但是与此同时，你已经失去了一个像我这样真正的客户。

(8) **区别对待（歧视）不同类型的客户。**如果我是一个生意人，我可能会使用老一点的操作系统，你不能因为我只是一个人，就强迫我升级到一个不稳定的新系统。如果我为了购买往返机票而浏览一家航空公司的网站，我希望它能一个不漏地列出所有航班，没准我还想选个联程往返呢。

(9) **提供了毫无用处的搜索功能。**我想搜索你的网站内容，不仅仅是搜索恰好与我的搜索关键词匹配的公关新闻稿。

(9.7) **把按键的标签弄错。**如果“浏览更多”按钮打开的是视频剪辑，我会很生气。我对这个按钮的理解是更多的文字信息，而不是坐在那里看你的演示。

不要让我讨厌你！



# 撰稿人

**Matteo Becchi，PMP（Arlington，Virginia，U.S.）**
**马泰奥·贝基，项目管理师（美国弗吉尼亚州阿灵顿市）**

马泰奥·贝基是一位项目管理师（PMP），喜欢寻找具有挑战性的工作。他有众多专业背景，如IT应用升级、强化和执行项目，以及包括技术更新、数据中心迁移和整合在内的硬件项目经验。马泰奥还具有管理直接报告和各种项目类型及利益相关群体的经验。同时，他对软件开发生命周期（SDLC）颇有研究，并且坚定支持PMI（美国项目管理协会）横跨九个知识领域的项目生命周期5个阶段的理论，他仍在不断追求改进和完善商业过程（项目管理领域内外）的方法，致力于把工作环境的效能和有效性提升到更高水平。

马泰奥最近在乔治华盛顿大学获得了信息系统和技术的硕士学位。目前，他参加了研究生水准的领导力建设认证学习。他还精通意大利语和西班牙语。马泰奥的邮箱是matteo.becchi@gmail.com。

**Cynthia A. Berg，PhD（ABD），PMP（Glendale，Arizona，U.S.）**
**辛西娅·伯格，在读博士，项目管理师（美国亚利桑那州格伦代尔市）**

辛西娅·伯格是伯格联合会计公司的所有人，也是领导力发展和项目管理专业的咨询顾问。她在美敦力公司（Medtronic）工作了20年，最近作为主要项目管理专家任职于坦佩（Tempe）的美敦力亚利桑那装备制造公司（Medtronic Arizona Device Manufacturing）。在亚利桑那，她主要从事新产品开发和项目管理培训及方法论的研究。她在任职期间，也为美敦力公司的生产、财务和新产品开发做出了贡献。

辛西娅是凯勒管理研究院（Keller Graduate School of Management）的资深教

师，从2000年起就在凯勒任教，教项目管理课程和工商管理课程的硕士研究生。她也在力拓社区大学（Rio Salado Community College）做兼职教授，为法人代表讲授商业和项目管理，被评为2000~2001学年杰出兼职教授。

辛西娅的教育背景包括：堪培拉大学的组织行为学博士，博士论文涉及裁员、幸存者综合症和员工激励；亚利桑那州立大学的工商管理硕士学位；圣凯瑟琳学院的心理学和社会学学士学位。

从1991年起，辛西娅作为美国项目管理协会的志愿者在国内和国际都非常活跃，1993取得了项目管理师（PMP）资格。辛西娅组织更新了*The Guide to the Project Management Body of Knowledge*（2000版），2001年获得PMI Linn Stuckenbruck年度人物大奖。1995年到1999年是项目管理协会成员，之后，2000年到2003年她参与了Standards MAG。目前，她是"项目管理威胁实践标准"项目的领导人。

她的其他兴趣包括水肺潜水（她是专业潜水教练协会的注册教练和潜水员）、健身运动、十字绣和刺绣、制作彩色玻璃，当然还包括看书。

### David Bock（Reston，Virginia，U.S.）
### 戴维·博克（美国弗吉尼亚州雷斯顿市）

戴维·博克2007年创建了CodeSherpas公司并任首席顾问。戴维也是北弗吉尼亚州Java用户组的主席、O'Reilly的OnJava.com网站的编辑，还经常在No Fluff Just Stuff Software Symposiums等知名软件交流大会上发表演讲。

2006年1月，戴维被Sun公司赞助的一个Java社区授予Java Champion称号，全世界获得该称号的活跃人士大约只有100名左右。

戴维也参与过几个JCP小组的工作，工作涉及Java 6平台和Java模块系统规范。

### David Diaz Castillo，MBA，PMP（Panama City，Panama）
### 戴维·迪亚兹·卡斯特罗，工商管理硕士，项目管理师（巴拿马巴拿马城）

戴维·迪亚兹·卡斯特罗在巴拿马的巴拿马城项目管理咨询公司任项目经理办公室主任。他做过政府部门和一些个人项目，涉及信息技术、法律和人力资源。他是一名系统工程师，还拥有财务方向的工商管理硕士学位和项目管理的研究生学位。他还在2007年获得了项目管理师（PMP）认证。

### Udi Dahan (Haifa, Israel)
### 尤迪·达罕(以色列海法)

尤迪·达罕是软件架构和设计方面的国际知名专家。因解决方案架构和连接系统连续四年被微软公司授予“最有价值专家”(MVP),尤迪也是微软公司的下一代技术平台顾问委员会中的一员,该委员会关注WCF/WF/OSLO、Software Factories Initiative(软件工厂计划)、Composite Application Library and Guidance。

尤迪是INETA(国际.NET联盟)承认的33位欧洲专家之一,是国际软件架构协会(IASA)可靠性、可用性和可扩展性方面的作者和讲师,是SOA、Web服务和XML方面的专家级人物,被世界上发行量最大的软件杂志*Dr. Dobb's*推荐过。除了咨询、培训和演讲之外,尤迪还领导着最受欢迎的开源.NET Enterprise Service Bus(.NET企业服务总线)项目NServiceBus。

小到互联网创业公司的Web项目(包括积极利用新技术的政府项目),大到投资数百万上千万美元、由几百名开发和测试人员参与的大型企业项目(涉及的企业千差万别,规模各异),都踊跃地委托尤迪根据他们当前及未来的需要,给出相应可靠的架构及设计建议。

可以通过尤迪的博客和他联系,地址是UdiDahan.com。

### Matt “Boom” Daniel (Coopersburg, Pennsylvania, U.S.)
### 马特·“布姆”·丹尼尔(美国宾夕法尼亚州库珀斯堡市)

马特·“布姆”·丹尼尔曾在美国海军陆战队担任战斗机飞行员、武器指挥官长达12年,毕业于海军战斗机武器学校(即大名鼎鼎的TOPGUN)。马特曾经指导和培训过很多军官,内容涉及小兵团领导能力、飞行作战策略、作战计划的制定和执行以及后勤保障。马特经常协调企业CEO和高层领导的对话,以期解决实际问题并制定相应的战略。1985年,马特获得弗吉尼亚军事学院土木工程专业理学士学位,2004年的1月,创立Business Battlefield公司,开始了他的企业领导和管理咨询生涯。马特有四个儿子,他和家人生活在宾夕法尼亚州的库珀斯堡。

**Barbee Davis，MA，PHR，PMP（Omaha，Nebraska，U.S.）**
**芭比·戴维斯，文学硕士，人力资源专家，项目管理师（美国内布拉斯加州奥马哈市）**

芭比是PMI *Community Post*中一个半月期专栏撰稿人，该专栏在全球有超过40万的项目经理订阅。她是一位国际评审员，审核那些希望获得或更新PMI的注册教育机构认证（REP）的培训组织。

芭比拥有一家计算机软件培训公司，致力于桌面和技术应用指导及认证，她是一位微软Project项目管理专家。实际上，她先前和别人一起写了一本书，即*How To Learn Microsoft Project in 24 Hours*。

芭比在美国为许多公司管理过项目，对数百个项目经理进行过认证培训，曾应邀到很多大学授课，是一位非常受欢迎的嘉宾演讲人。没准你就在美国或加拿大的No Fluff Just Stuff大会讲台上见过她。

**Monte Davis，MCSE（Omaha，Nebraska，U.S.）**
**蒙特·戴维斯，微软认证系统工程师（美国内布拉斯加州奥马哈市）**

自从2006年3月加盟到微星科技，蒙特·戴维斯先后负责Windows服务器管理、IT项目管理、服务器备份和恢复、电子邮件管理、新服务器展示、局域网（广域网）管理和二级支持。蒙特熟练掌握了各个版本的Windows服务器管理、思科公司IOS操作系统、Linux操作系统、Lotus Domino、Microsoft Exchange 2003和TCP/IP协议。

蒙特原来在Retalix（即先前的Integrated Distributed Systems）工作，在那里他做了5年的网管部经理和网站管理员。他还在易得训（ExecuTrain）公司做过5年的网络管理员，管理过服务器、LAN/WAN以及电子邮件系统。他还讲授过Microsoft Official Curriculum课程。

蒙特有西北密苏里州立大学的学士学位，是VMware用户组成员。

他通过的认证包括 MCSE（Windows 2000、Windows Server 2003）、MCSA（Winodws NT 4.0、Windows Server 2003）和 MCP 2.0。

**Scott Davis（Broomfield，Colorado，U.S.）**
**斯科特·戴维斯（美国科罗拉多州布鲁姆菲尔德市）**

斯科特·戴维斯2006年通过Grails率先实现了一个公共网站，从那时起，他

就一直积极致力于该项技术的研究。

他是 *Groovy Recipes:Greasing the Wheels of Java and GIS for Web Developer*（Progmatic）等技术专著的作者，也为 IBM developerWorks 撰写过两组系列文章（“Mastering Grails” 和 2009 年的 “Practically Groovy”）。斯科特认为 Groovy 和 Grails 代表 Java 的未来，为此他一直笔耕不辍。

斯科特通过公开课和内部培训的形式为创业公司及财富100强公司讲授Groovy和Grails课程。他是Groovy/Grails Experience大会的共同创办人，是各种国际技术会议上的常客（包括No Fluff Just Stuff、JavaOne、OSCON、TheServer-Side和QCON）。

2008年，斯科特因为在JavaOne的演讲“Groovy, the Red Pill: How to blow the mind of a buttoned-down Java developer”，被投票选为最佳演讲人。

可以通过斯科特的公司ThirstyHead（thirstyhead.com）联系他。

**Neal Ford（Atlanta，Georgia，U.S.）**
**尼尔·福特（美国佐治亚州亚特兰大市）**

尼尔·福特是ThoughtWorks的软件架构师和技术培养者（Meme Wrangler）。ThoughtWorks是一家全球化IT资讯公司，专注于端到端的软件开发和交付。他也是应用程序设计和开发人员，经常编写教材，为杂志写文章，做课件和视频/DVD展示片。尼尔还是5本不同技术领域图书的作者/编辑。他的研究重点是设计和构建大规模企业级应用程序。

尼尔还是一位享有国际声誉的演讲人，在世界上100多个开发人员会议上做过演讲，并参与了600余次的对话。欢迎访问他的网站http://www.nealford.com，并通过nford@thoughtworks.com联系他。

**Jorge Gelabert，PMP（Berlin，Connecticut，U.S.）**
**乔治·格拉伯特，项目管理师（美国康涅狄格州柏林）**

乔治·格拉伯特是经过认证的项目管理师，任美国东北电力公司（Northeast Utilities）项目经理。他拥有康涅狄格州桥港大学（Bridgeport）的计算机工程学士学位，拥有伦斯勒理工学院（Rensselaer Polytechnic Institute）计算机科学硕士学位，能说西班牙语（母语）和英语，同时还是Delphi DMP，拥有乔治华盛顿

大学颁发的项目管理方向的Master和Advanced Master认证证书。

乔治是PMI的活跃成员，曾任市场总监、会长及PMI新英格兰地区南部分会会长等职。当前，他担任第三区Component Mentor，支持19个PMI分会。他也是PMI-ISIG和IEEE成员。

**Dr. Paul Giammalvo，CDT，CCE，MScPM（Jakarta，Indonesia）**
**保罗·贾马尔沃博士，CDT，CCE，MScPM（印度尼西亚雅加达）**

保罗博士是印尼雅加达PT Mitratata Citragraha（PTMC）公司（www.getpmcertified.com）的高级技术顾问（项目管理）。他还是法国巴黎里尔高等商学院（www.esc-lille.com）的项目及计划助理教授，在南亚和东亚、中东和欧洲从事项目管理培训及咨询工作长达14年之久。

他积极活跃于全球的项目管理社区，为全球项目管理从业人员贡献自己的力量。为此，他作为专家积极参与AACEI（国际成本工程进展协会）、CSI（建筑规范学会）和CMAA（美国建筑管理协会）的活动。他也是位于澳大利亚悉尼的GAPPS（项目绩效标准国际联盟，www.globalpmstandards.org）的董事，并为位于伯斯的西澳大学（www.blendedlearning.ecm.uwa.edu.au）制定资产和项目管理专业的研究生教学大纲。

保罗博士最近 35 年中有 18 年一直致力于大规模、高技术含量的国际项目，其中包括获得了国际赞誉的大项目，如以色列澳华特的内盖夫空军基地项目（《戴维营和平协议》的一部分）和阿拉斯加管道和远程早期预警点项目的升级。最近的大多数时间，他为苏门答腊岛 Caltex Minas 油田项目做成本与计划咨询顾问，并担任雅加达 Bakrie 兄弟公司 Taman Rasuna 公寓综合大楼项目的项目经理。他服务过的公司和机构包括 AT&T、爱立信、诺基亚、朗讯、通用汽车、西门子、康诺克石油公司、优尼科、英国石油公司、Dames and Moore、兰万里、Freeport McMoran、德士古石油公司、联合国项目机构、世界银行研究院以及许多跨国公司和非政府机构。

保罗拥有乔治华盛顿大学的建筑管理学士学位和项目管理硕士学位。在Christophe Bredillet博士（CCE，IPMA A Level）的指导下，他最近在里尔高等工业管理学院和里尔高等商学院取得了项目和计划管理的博士学位。

### Karen Gillison (Leesburg, Virginia, U.S.)
### 卡伦·吉利森（美国弗吉尼亚州利斯堡市）

卡伦·吉利森拥有深厚的计算机科学背景，有15年跨行业和跨技术研发和交付软件系统的经验。卡伦为商业和政府客户提供软件开发和项目管理服务。

卡伦是RubyNation组织委员会、北弗吉尼亚Ruby用户组和Java用户组成员，经Pragmatic Studio Ruby on Rails培训获得Golden Spike毕业生称号，2000年和2006年获得了FGM技术成就奖。

卡伦和她的丈夫查尔斯、两个孩子以及两只拉布拉多猎犬居住在弗吉尼亚州。

### James Graham, PMP (Ta' l-Ibrag, Malta)
### 詹姆斯·格雷厄姆，项目管理师（马耳他塔尔艾布安戈市）

詹姆斯·格雷厄姆是独立管理顾问，足迹遍及全球。

他的研究领域涉及人力资源开发、设计和交付管理开发计划，以及业务流程改进及公司文化咨询。

詹姆斯拥有项目管理师的资格证书，同时也是分析师及程序员学会（IAP）的理事，有咨询硕士学位和心理学学士学位。

他居住在欧洲南地中海的马耳他岛上。

### Alan Greenblatt (Sudbury, Massachusetts, U.S.)
### 艾伦·格林布拉特（美国马萨诸塞州萨德伯里市）

作为Sciova的创始人和CEO，艾伦·格林布拉特在这一角色中倾注了自己25年的软件开发、技术管理经验和创业精神，Sciova是一家专门开发企业级语义应用的公司。作为Metatomix公司语义技术副总裁，艾伦在该公司历经几年的高级语义技术平台开发过程中充当了举足轻重的角色，并拥有与这项工作相关的专利。他经常打交道的客户包括空中客车公司、道富银行（State Street Bank）、优时比（UCB）制药公司，帮助这些客户开发独特的具有很高价值的语义应用。多年以来，艾伦对Sun和微软提供的高级媒体技术发展也做出过重要的贡献。他还与别人合伙创立了软件咨询公司Anyware Fast，并在一家很早的Java创业公司Dimension X中做过销售。Dimension X的业务是开发虚拟现实和交互式多媒体创作软件。艾伦在Dimension X曾担任技术主管，直到该公司被微软收购。在成为

微软Office团队的一员后，他领导了微软Vizact 2000的开发。Vizact 2000是一款交互式多媒体创作工具。

艾伦拥有滑铁卢大学电气工程应用科学的学士学位。

Kim Heldman，PMP（Lakewood，Colorado，U.S.）
**金·海德曼，项目管理师（美国科罗拉多州莱克伍德市）**

金·海德曼拥有18年的IT项目管理经验。她是科罗拉多州运输部的CIO，负责一个拥有分布在各地的3300多名员工及10亿美元的组织的全部IT资源规划、预算编制以及项目管理工作。

金曾担任多年行政领导职务，公认具有战略远见，擅长与各种团队和组织协同工作，进行目标激励，鼓舞士气，并领导自己的团队达成难以置信的目标。

金在管理政府部门的各种规模和范围的项目方面经验丰富。她是州长信息技术办公室行政管理委员会成员，负责监管和管理由州政府批准的IT项目。作为科罗拉多CIO论坛的联合主席，她帮助推荐和评审全州的信息技术政策、标准和活动。金也是科罗拉多项目管理用户组的共同创办人，该用户组向全州各公司CIO及IT项目经理开放。她对州一级的项目管理政策、标准和办法的制定及实施做出了贡献。这个用户组在推动立法方面也起到了重要作用，通过立法确保了科罗拉多州所有重要的IT项目都能由通过认证的项目经理来管理。

金是畅销书 *PMP® Project Management Professional Study Guide* 的作者，该书由 Sybex 公司出版，现在已经是第 4 版了。世界上有成千上万的人在使用这本书，为成功通过项目管理师考试做准备。金也是 *Project Management JumpStart*（第 2 版）和 *Project Manager's Spot Light on Risk Management*（两本书都由 Jossey-Bass 公司出版）的作者。她还是 *Excel 2007 for Project Managers* 和 *PMP® Project Management Professional Study Guide*（精装版，Sybex 公司出版）的合著者。

金还写了很多关于领导力的文章，并在众多会议和活动上演讲。她和丈夫比尔还有三个未成年的孩子一起居住在科罗拉多州的阿瓦达。

Naresh Jain（Malad，Mumbai，India）
**纳雷什·贾因（印度孟买市马来德）**

纳雷什·贾因是一名程序员，在Directi公司负责技术布道。他运用敏捷和精

益思想帮助软件组织交付高质量的软件解决方案。他从2003年开始在各种软件项目中运用XP、Scrum和 Crystal技术。

纳雷什致力于建设由大批有能力、有天赋的程序员组成的社区，塑造印度软件行业的下一代领军人物。他在印度建立了敏捷技术用户组并创办了Simple Design and Testing大会（SDTConf），为了表彰他的成就，敏捷联盟于2007年授予纳雷什戈登·帕斯克奖（Gordon Pask Award）。

纳雷什是印度敏捷软件社区（ASCI）的创建者和副主席。他组织了形式多样的会议，包括SDT大会、敏捷教练训练营。在印度，纳雷什支持建立了各种敏捷技术用户组，包括Agile Philly 用户组等。

纳雷什是开源代码的积极提交者，喜欢在大学教授软件开发课程。纳雷什也积极帮助软件公司使用敏捷技术。

纳雷什喜欢啤酒、音乐、探险运动和各种颜色的辣食。他的邮箱是：naresh@agilefaqs.com。

**Krishna Kadali，M. Tech（Kondapur，Hyderabad，India）**
**克利斯那·卡达利，工科硕士（印度海得拉巴市康达巴）**

克利斯那·卡达利在技术及针对业务开发技术方案方面拥有超过19年的实践经验。克利斯那曾经在一家高科技创业公司（也是一家大众贸易公司）工作，他能够根据软件产品制定出成功的业务方案。目前，他在印度海得拉巴领导一家系统和数据集成服务供应商Prabhavat Solutions，这家公司能提供一些系统和数据整合解决方案。

在创建Prabhavat之前，克利斯那是位于弗吉尼亚麦克莱恩的Nimaya公司的创始人和CTO，在那里他领导开发了公司的旗舰产品ActionBridge和InSync，也交付了其他几个用户数据和系统集成方案。作为Nimaya的分部，他在印度组建了一支海外团队，通过这支团队，他成功交付了几个技术方案。

在创建Nimaya之前，克利斯那在弗吉尼亚费尔法克斯的MKS以及法国巴黎的BULL S.A.公司工作，主要负责架构、设计、研发，并交付了旗舰产品NuTCRACKER和OSIAPI，积累了丰富的编程经验。

克利斯那拥有位于印度克勒格布尔（Kharagpur）的印度理工学院的通信硕士学位，以及位于印度阿嫩德布尔（Anantapur）的贾瓦哈拉尔·尼赫鲁技术大

学的电子和通信学士学位。

Patrick Kua（London，UK）
**帕特里克·夸（英国伦敦）**

帕特里克·夸是ThoughtWorks的敏捷技术教练、推动者和开发人员。最近四年，他一直与不同团队成员在敏捷环境下一起工作，清楚在协作方式下人们的工作是多么有力和高效。他一直喜欢研究持续改进和通过简化流程来提高团队效率。他既能提升团队的技术水平，也对团队管理流程有深刻的理解，这两点都能帮助团队迈向成功。

Anupam Kundu（New York，New York，U.S.）
**阿努潘·昆杜（美国纽约州纽约）**

阿努潘·昆杜是ThoughtWorks的项目经理（主要顾问），大部分时间在纽约。他有多个行业的咨询背景，也具有不同业务领域的大型项目管理经验，比如HR、投行内部网、私有股份公司、北美电信BACC、全球出版与媒体、老年健康护理系统以及产权保险。阿努潘拥有管理知名企业软件项目各个阶段（搜集需求、评估、分析和设计、实施、质量控制、培训和实施后活动）9年多的经验。

Lukeman Lawal，M.ENG，MNSE，R.ENGR.（Lekki，Lagos，Nigeria）
**卢克曼·拉瓦尔，工程学硕士，MNSE，R.ENGR（尼日利亚拉各斯市莱基）**

卢克曼·拉瓦尔是尼日利亚雪弗龙有限公司的项目工程师。他管理石油和天然气项目，包括工程设计、建设和安装。卢克曼还是拉沃斯河海上油气 3A 项目（EGP3A）和新油田领域开发项目工程师。他目前是一名前期概念开发项目工程师。

卢克曼是尼日利亚贝宁城贝宁大学的机械工程学院的老师。

Martha Legare，MBA，PMP（Atlanta，Georgia，U.S.）
**玛莎·勒加雷，工商管理硕士，项目管理师（美国乔治亚州亚特兰大市）**

玛莎·勒加雷在北美和欧洲任教练、培训师和咨询顾问长达20年。她是Gantt集团的CEO。Gantt集团是一家咨询和培训公司，业务涉及战略性规划、项目管理和行为科学。Gantt集团的客户主要集中在营销和广告产业。

玛莎设计并主讲了多期研讨班，对象是罗约拉大学（Loyola University）MBA学位的项目管理资格认证学员，她是罗耀拉大学的助理教师。她的文章刊登在美国经营管理协会《训练员活动书籍》上。玛莎已经写了项目方法论，并为美国、墨西哥和欧洲的发展项目办公室提供帮助。

玛莎是注册项目管理师、美国仲裁协会的注册调停人。她获得了阿尔梅达大学的MBA学位和基尔福特学院的文学学士学位。她天生具有战略思想家的潜质，不仅具有多种文化沟通能力，而且对改善客户的业务也坚忍不拔。

**James Leigh（Toronto，Ontario，Canada）**
**詹姆斯·利（加拿大安大略省多伦多市）**

詹姆斯·利是一位独立软件顾问，有10年的Web解决方案设计经验，住在多伦多。他能娴熟地通过软件对业务问题和概念建模，对性能和技术整合有独到研究。他的专业背景包括语义网技术和分散式计算机网络。

詹姆斯也领导过与语义相关的开源软件项目，包括Sesame的联合RDF商店、关系RDF商店、Sesame的服务器客户端库、Mulgara与Sesame的集成以及OpenRDF的对象RDF映射程序。他还投入5年多时间领导优化过不同的Java应用，包括Sesame的RDF商店的基准测试和优化、AMC剧院员工排班系统和泳装设计师Christina及Gottex的制造跟踪系统。

**Craig Letavec，PMP，PgMP，MSP（Waynesville，Ohio，U.S.）**
**克雷格·莱特维克，项目管理师，项目集管理专业人士，管理成功计划人士（美国俄亥俄州威恩斯维尔市）**

克雷格·莱特维克曾在不同规模的公司里担任项目、程序管理经理和公司的项目管理办公室经理，这些公司包括宝洁&甘布尔、休利特帕卡德和西门子股份公司。他的经历包括领导全球软件开发和项目的实施、领导项目管理办公室、开发和讲授项目管理训练课程。克雷格拥有项目管理师、项目集管理专业人员的资格证以及管理成功计划认证，也是乔治华盛顿大学的项目管理理科硕士。著有*The Program Management Office:Establishing: Managing, and Growing the Value of a PMO*——一本有关PMO开发和管理的畅销参考书，以及*Program Management Professional (PgMP)®: A Certification study Guide with Best Practices for Maximizing*

*Business Results*（均由J. Ross出版社出版）。

Randy Loomis，PMP（Andover，Connecticut，U.S.）
**兰迪·卢米斯，项目管理师（美国康涅狄格州安多弗市）**

兰迪·卢米斯是一位拥有14年项目管理经验的注册项目管理师，已经在IT领域工作了26年。他现在是东北电力公司IT项目集管理办公室的项目经理。兰迪以优秀的成绩从东康涅狄格州立大学毕业，获得心理学学位。

Kim MacCormack（Leesburg，Virginia，U.S.）
**金·麦科马克（美国弗吉尼亚州利斯堡市）**

金是软件咨询公司CodeSherpas的创始人之一，通过这家公司她把自己的软件工程经验传递给了商业和政府客户。工作中，她致力于开发高质量的Web应用，帮助她的客户将他们的产品更快地投入市场。

金有超过13年的软件和系统工程的经验。她设计和开发过各种各样的基于网络和客户端/服务器应用程序。2005年，她获得了软件工程方面的硕士学位。金也是较早使用Ruby on Rails的人，是Pragmatic Studio Ruby on Rails培训课程的Golden Spike毕业生。另外，她是北弗吉尼亚州Java和Ruby用户组的成员。

金在软件工程生命周期的每个阶段都很有经验，包括需求分析、设计、人机界面设计、代码开发和验证。作为项目经理、技术领导及软件工程师，她完成过30多个Web应用，涉及商业、非盈利和政府客户。这些项目包括网站、企业内部网、电子商务应用软件、内容管理系统以及复杂的基于Web的调查工具，支持创建多语言的用户调查。

Kathy MacDougall（Erie，Colorado，U.S.）
**凯西·麦克杜格尔（美国科罗拉多州伊利）**

凯西·麦克杜格尔是Zepheira公司的首席业务架构师，Zepheira提供的解决方案用于有效地整合、利用和管理个人、小组及企业范围内的数据。作为一名架构师，凯西负责分析顾客现有的业务模型，为项目改进提供建议，确保成功采用新的技术实现。

凯西在调动整个企业的积极性帮助公司进行评估、管理和利用公司数据，增

加收入、发掘新商业智能方面有丰富的经验。在这个领域，她在20年工作生涯中取得了很多骄人的业绩，包括为5到100亿美元的公司创建基于数据的知识管理系统，其中就有通用电气和Sun等公司。

从2000年起，凯西利用语义网技术实施解决方案。在Sun公司，凯西和她的团队领导了前所未有的一次语义技术项目的大规模实施，这个项目为组织内部的动态内容分发打下了基础。有了技术和商业基础设施的支持，Sun公司每年可以节约开支上千万美元。2003年，凯西和他的团队应邀请在W3C成员大会上介绍了在商业方案中应用语义Web技术的经验。

凯西毕业于科尔盖特大学，在流程改进和组织变革方面接受了大量培训，包括Six Sigma方法论。

**Ernani Marques da Silva，MBA，PMP，PgMP（Mairipora，Sao Paulo，Brazil）**

**埃尔纳尼·马奎斯·达席尔瓦，工商管理硕士，项目管理师，项目集管理专业人员（巴西圣保罗州迈里波朗市）**

埃尔纳尼·马奎斯拥有超过19年的涉及面极广的成功经验，包括IT、银行和服务业的项目管理、程序管理及组合管理等。他领导和管理过很多PMO（项目管理办公室）以及其他多个项目和团队。

他在IT领域的经验包括项目管理和程序管理，领域涉及应用研发、产品管理，以及利用SDLC（系统开发生命周期）和项目管理方法论进行系统整合。

**Alex Miller（Ballwin，Missouri，U.S.）**

**亚历克斯·米勒（美国密苏里州巴尔温市）**

亚历克斯是Terracotta公司的技术领导和工程师，Java开源平台Terracotta就是该公司的产品。在加入Terracotta之前，亚历克斯在BEA Systems参与AquaLogic产品线的工作，是MetaMatrix的首席架构师。他的兴趣涉及Java、并发、分发系统、查询语言和软件设计。

亚历克斯非常喜欢写他在 http://tech.puredanger.com 上的博客。他曾和 Terracotta 团队的其他成员于 2008 年合著出版了 *The Definitive Guide to Terracotta* 一书（Apress）。亚历克斯还经常在 Java 用户组会议、No Fluff Just Stuff 和 JavaOne 之类的会议上演讲。

William J. Mills（Castro Valley，California，U.S.）
**威廉·J.米尔斯（美国加利福尼亚州卡斯特罗谷）**

威廉·J.米尔斯目前在雅虎公司负责软件和产品安全。此前，他在Invisible Worlds公司（已不存在）工作过。在富国银行工作时，他负责防火墙，并参与了第一个网络银行版本的发布。他还承接过各种零星工作，并曾在圣迭戈高级法院任过职。

Gennady Mironov，CPM（Toronto，Ontario，Canada）
**杰纳迪·米罗诺夫，CPM（加拿大安大略省多伦多市）**

杰纳迪·米罗诺夫出生于1967年。他在苏联军队服役两年，1992年毕业于莫斯科动力工程技术大学，获得电力工程硕士学位。2001年获得心理学和商业研究生学位。

过去的16年里，杰纳迪在IT和通信业扮演过从技师到解决方案经理、再到程序经理的角色。在他移居加拿大之前的4年里，他基于西门子和华为的解决方案，在俄罗斯管理过很多无线通信项目。由他直接负责的项目金额高达1600万美元。

2008年，杰纳迪在多伦多的汉伯应用技术学院完成了为时1年的项目管理研究生学习。他最近在向PMI申请项目管理师考试。

他和他的妻子以及三个孩子住在多伦多。

Jared Richardson（Morrisville，North Carolina，U.S.）
**贾里德·理查森（美国北卡罗来纳州莫里斯维尔）**

贾里德·理查森是*Ship It! A Guide to Successful Software Projects*[①]（Pragmatic）的合著者，这本书已经被翻译成4种语言。他与NFJS One一起为各种团队和经理提供咨询。可以在http://AgileArtisans.com和http://NFJSOne.com上联系到他。

Brian Sam-Bodden（Scottsdale，Arizona，U.S.）
**布赖恩·萨姆-博登（美国亚利桑那州斯科茨代尔）**

布赖恩·萨姆-博登是一位作家和国际知名的演说家，在一些财富500强公

① 本书中文版将由人民邮电出版社出版。——编者注

司里担任过架构师、开发人员、指导员和培训师。他是*Beginning POJOs: Spring, Hibernate, JBoss and Tapestry*（Apress）一书的作者，还与人合著了*Enterprie Java Development on a Budget: Leveraging Java Open Source Technologies*。

**Angyne J. Schock-Smith，PMP（Phillipsburg，New Jersey，U.S.）**
**安吉妮·肖克－史密斯，项目管理师（美国新泽西州菲利普斯堡）**

安吉妮是Arysta Projex公司的前总裁和首席执行官，该公司是一家独立公司，已经为项目管理社区服务了10年之久。在此之前，安吉妮在多个行业里都担任过项目经理职务，从事的工作包括提供全球网络解决方案、销售支持、客户关怀、产品管理和战略计划等，她还在AT&T工作了18年。2008年10月1日，她成为国际学习协会（IIL）全球学习解决方案（Global Learning Solutions）项目的高级教学设计师，专门从事项目（程序）管理、领导力培训和咨询服务。

**Matt Secoske（Omaha，Nebraska，U.S.）**
**马特·萨克斯科（美国内布拉斯加州奥马哈）**

马特·萨克斯科是Nimblelogic公司的技术骨干，Nimblelogic公司专注于开发精品Web应用程序。他的博客是 http://mattsecoske.com，Twitter账号是@secos。

**Richard Sheridan（Ann Arbor，Michigan，U.S.）**
**理查德·谢里登（美国密歇根州安阿伯）**

在经商仅仅2年之后，谢里登，Menlo Innovations公司的CEO，就成为《福布斯》“雇佣自己”封面故事的主角，为那些选择创业而不是失业的人树立了榜样。一年后，《华尔街日报》的一篇文章又报道了Menlo用来设计和开发软件的办公室。6年间，Menlo一跃成为美国500家发展最快的私人公司之一。更值得一提的是，他的故事是在IT产业人心不定、人们纷纷尝试寻求项目外包的大背景下发生的。

谢里登领导的Menlo团队打破陈规，却为客户创造了非同寻常的成果。在新泽西州门洛帕克的爱迪生发明工厂里，没有墙、没有办公室、没有门，也没有隔间——就是一个大的开放的房间。在这种嘈杂、活跃的气氛中，Menlo开发的软件被运用到了各行各业，从卫生保健到科学仪器，从时尚的电子商务到柴油发动机汽车检修，还有更多的其他领域。

Menlon为客户开发的软件由Menlon的High-tech Anthropologists®（高技术人类学家）针对普通人设计，由Menlon的世界一流敏捷软件开发团队着眼于长远而构建，由位列项目管理协会评出的美国50个最多产的专业项目管理经理中的知名项目经理管理。谢里登和Menlon赢得了很多的奖项和荣誉，他和他的团队经常被邀请出席国内和国际会议，向人们分享Menlon Software Factory™创建学习型组织、从而紧跟软件和设计进步步伐的经验。

Derry Simmel，PMP，MBA，FLMI（Chapin，South Carolina，U.S.）
**德里·齐美尔，项目管理师，工商管理硕士，FLMI（美国南卡罗来纳切宾市）**

德里·齐美尔具有在IT和项目管理领域25年的从业经验。最近，他一直致力于组建PMO（项目管理办公室），过去的6年里就成功组建了3个。其中最后一个负责管理南卡罗来纳政府的一个价值8900万美元的项目。德里有菲尼克斯大学的MBA学位和南卡罗莱纳州大学的计算机科学学士学位。他现在担任PMI项目管理办公室特别兴趣小组副主席以及PMI Midlands分会的程序副会长。

Pavel Simsa，PMP（Bellevue，Washington，U.S.）
**帕维尔·西姆沙，项目管理师（美国华盛顿州贝尔维尤市）**

帕维尔·西姆沙在软件开发和本地化业务领域已经工作了10年，其中有5年从事企业安全软件项目管理。他为一个国际性的公司工作，公司每一个项目的利益相关者都遍布全球。每一个产品一般都要支持10～17种语言。尽管他到2008年才得到他的项目管理师证书，但他已经根据PMBOK® Guide实践了好几年，努力将它们运用到独特、灵活而又富有挑战性的软件开发工作中。

Ken Sipe（St. Charles，Missouri，U.S.）
**肯·赛普（美国密苏里州圣查尔斯市）**

肯·赛普是Perficient（PRFT）公司技术总监。他曾经是CodeMentor公司的创建人，在CodeMentor公司任首席架构师和顾问，在RUP执行期间和在软件解决方案交付过程中运用敏捷技术方法论时引导客户。他是Rational在OOAD和RUP方面的前任培训师，还在Borland公司任 CORBA Visibroker培训师。他现在仍非常喜欢为软件开发的各个方面提供培训和指导。

肯也经常在No Fluff Just Stuff大会上演讲。

Marty Skomal，MPA（Omaha，Nebraska，U.S.）
**马蒂·斯科莫，公共管理硕士（美国内布拉斯加州奥马哈市）**

马蒂·斯科莫在内布拉斯加州艺术协会任项目总监，他负责监管所有的大型项目，包括艺术教育、多文化教育以及艺术旅游。他担任会议主持人，并为众多国家艺术机构的特别小组成员，如肯尼迪表演艺术中心和全国艺术基金会。他曾通过艺术管理奖学金计划获得过NEA（国家教育协会）的奖学金资助，并曾担任过国家艺术项目评估师和顾问。

马蒂拥有内布拉斯加州大学的公共管理硕士学位。

Brian Sletten（Beverly Hills，California，U.S.）
**布莱恩·斯莱滕（美国加利福尼亚州比佛利希尔斯市）**

布莱恩·斯莱滕是一个文科出身的软件工程师，关注新技术。他曾做过系统架构师、开发人员、顾问和培训师。他的工作经验很广，涉及在线游戏、安防、财政以及商业领域的安全、网络矩阵开关控制、三维模拟（可视化）、网格计算、P2P和以语义网为基础的系统。他拥有威廉玛丽学院的计算机科学学士学位，现在住在加州的比佛利山庄。他现在位于加州卡尔弗城的 Riot Games公司任高级平台工程师，参与《英雄联盟》（League of Legends）的开发。他重点关注Web架构、面向资源的计算、语义Web、可扩展系统以及安全问题。

Venkat Subramaniam（Broomfield，Colorado，U.S.）
**温卡特·苏布拉马尼亚姆（美国科罗拉多州布鲁姆菲尔德市）**

温卡特是Agile Developer公司的创始人，曾培训和指导过来自美国、加拿大、欧洲和亚洲的数千名软件开发人员。他使用敏捷开发技术和其他多种软件技术帮助客户成功。

温卡特经常被各种国际软件会议邀请做演讲。他写过 *.NET Gotchas* 一书（O'Reilly），还与人合著了 *Practices of an Agile Developer*[①]（Pragmatic Bookshelf）。他的新书 *Programming Groovy* 也是由 Pragmatic 出版的。

① 中译本《高效程序员的45个习惯：敏捷开发修炼之道》由人民邮电出版社出版。——编者注

通过venkats@agiledeveloper.com可以与他联系。

**Miyoko Takeya，PMP（Tokyo，Japan）**
**竹家美代子，项目管理师（日本东京）**

竹家美代子是PMI、PMAJ（PMI日本分会）和日本SEMS（软件工程及管理学会）会员。

她在日本的IT行业工作了30多年，开始时是日立公司的一名操作系统程序员，之后到DEC公司任职，后来又在日本NCR工作。

在DEC和NCR公司工作期间，美代子在管理项目方面投入了大量时间，并推动了关乎业务质量和绩效的几个项目。她组建过一个PMO，通过此办公室她实现了项目管理系统、项目记账系统、项目定价系统、活动报告及跟踪系统，以及许多其他成功改善业务质量和绩效的系统。

美代子非常喜欢她在IT行业项目领域的工作。目前，她在经营自己的项目管理咨询公司。

**Fabio Teixeira de Melo，PMP（Coatzacoalcos，Veracruz，Mexico）**
**法比奥·特谢拉·德梅洛，项目管理师（墨西哥韦拉克鲁斯州夸察夸尔科斯市）**

法比奥·特谢拉·德梅洛在欧德布莱克特公司任计划管理经理，该公司是巴西跨国公司欧德布莱克特集团的建筑分公司，集团总部设在巴西萨尔瓦多，并在15个国家设有分支机构。法比奥在建筑领域已有15年的工作经验，涉及能源、石油、天然气和石油化工等领域的EPC项目。他是领导学院（Leadership Institute）2004级的毕业生，还是PMI累西腓市、伯南布哥州以及巴西分会的创始人及前任会长。法比奥参与了*PMBOK® Guide*之Construction Extension部分和*Practice Standard for Scheduling*®的撰写工作，曾任DPC-SIG拉丁美洲主席，任期5年。

**Luis E. Torres，PMP（San Rafael，Alajuela，Costa Rica）**
**路易斯·托雷斯，项目管理师（哥斯达黎加阿拉胡埃拉省圣拉斐尔市）**

路易斯·托雷斯是PMI认证的项目管理师。他拥有西班牙巴塞罗那拉曼·鲁尔大学的项目管理硕士学位、哥斯达黎加大学银行和金融专业及国际商务的MBA学位以及机械工程专业的证书。路易斯在战略计划和预算、项目管理、跨国公司

的财务分析、国际程序合同管理以及项目工程领域有着超过15年的工作经验。

Harry Tucker（Matawan，New Jersey，U.S.）
**哈里·塔克（美国新泽西州马塔瓦恩市）**

哈里·塔克（http://www.harrytucker.com）一心在领导能力、协作和个人展示方面提倡卓越。他认为现在的社会、政治和环境条件保证了紧迫感的存在，从而促使他在人们身上培养这些领导特质。对于这一点，他和他的同事与在个人展示和领导力方面非常优秀的领导一起工作，以便能够让他人具备相应的技能和知识，同时激发他们为世界做贡献的热情。

哈里现在作为领导力培养者和战略顾问服务于财富100强公司，并为华尔街客户服务了近20年。之前，哈里担任过高级企业战略顾问和微软的架构师。他是微软个人展示小组（Microsoft Personal Empowerment Group）的创建人，该小组是微软公司内部的一个团体，致力于个人成长和职业成功。2005年，哈里还为城里的年轻人制定了一个目标既定的人生计划。

除了与妻子罗文以及三个可爱的孩子一起享受生活外，哈里还喜欢钓鱼、阅读、写作、学习、演讲和讲授个人展示课。

Lorin Unger（Hoboken，New Jersey，U.S.）
**洛林·昂格尔（美国新泽西州霍博肯市）**

洛林·昂格尔在技术策略和管理方面已有超过12年的工作经验，工作环境从网站到金融领域。

他的特长包括技术策略、队伍建设和管理、过程创建和执行、离岸外包计划执行和管理、效益分析和极强的包容心。

Angelo Valle（Rio de Janeiro，Brazil）
**安杰洛·瓦尔（巴西里约热内卢）**

安杰洛·瓦尔是一位技术革新能手、土工程师，拥有巴西里约热内卢联邦大学的建筑管理硕士。他还是上一任PMI里约热内卢分会的会长。

安杰洛还是一位知名作家，撰写了大量文章。他最近对PMO和挣值（earned value）非常感兴趣。作为瓦格斯基金会MBA的学术协调人，目前他正在负责2万

多名研究生的教育工作。

**Lelio Varella，PMP（Tijuca，Rio de Janeiro，Brazil）**
**莱利奥·瓦芮拉，项目管理师（巴西里约热内卢州蒂茹卡市）**

莱利奥·瓦芮拉是一位有着30多年工作经验的商业管理顾问，他重点关注战略规划和组织发展，组合、程序、项目管理，项目管理办公室。他为一些巴西最重要的公司工作，这些公司涉及IT、石油和天然气领域。作为一名经验丰富的代表性人物和指导人员，他已经参与过三本项目管理书籍的编写。莱利奥十多年来一直积极为PMI做义工，他的成就包括在里约热内卢建立PMI分会，现有成员逾1000人。

**Paul Waggoner，MBA，PMP，MCSE，CHP，CHSS（Waukee，Iowa，U.S.）**
**保罗·瓦戈纳，工商管理硕士，项目管理师，微软认证系统工程师，综合业务处理系统工程师，公共卫生信息系统工程师（美国爱荷华州沃基市）**

保罗·瓦戈纳是独立项目管理顾问和约聘项目经理。保罗在卫生保健、信息技术、安全领域已有超过20年的工作经验。作为一名卫生保健领域的专家，他的工作囊括了采购和供应两个方向。

在过去的10年里，保罗作为项目经理，与他们共同组建了一个PMO，完成了各类系统和临床项目。他还曾担任过不同的技术和管理职位，主管过中西部地区一个大型信息系统部门。他还是一个计算机培训公司的合伙人，负责过技术和管理工作。

**Adrian Wible（New York，New York，U.S.）**
**阿德里安·威布尔（美国纽约州纽约市）**

阿德里安·威布尔的自选头衔是“软件开发的催化剂”，他为ThoughtWorks公司工作，大多数时间担任项目管理的角色，但是，为尽力避免“脱离技术”，他经常亲自参与软件开发。他在IBM和戴尔电脑公司先后工作了20多年，在IBM做开发人员时，他接受了瀑布和SDLC软件开发模式，然后又转到做项目、人事和过程管理的工作。2005年，阿德里安加入了ThoughtWorks公司，并发现了敏捷宣言（XP、Scrum等），他意识到项目工作和管理一样也会有乐趣、刺激和收获。从那以后他就一往无前了。

联系阿德里安的方式是awible@thoughtworks.com。

**David Wood（Fredericksburg，Virginia，U.S.）**
**戴维·伍德（美国弗吉尼亚州弗雷德里克斯堡市）**

戴维·伍德是Zepheira公司的合伙人，他负责管理软件项目和推荐突破性的技术应用，以帮助客户获得最多的商业机会。

自从 1999 年开始，戴维就开始参与语义 Web 标准、工具、产品和服务的开发中。 他在 W3C 担任语义 Web 最佳实践和部署工作小组的联合主席，是语义 Web 协调小组的成员。他还是 Kowari Metastore、Mulgara Semantic Store 和最近重新设计的 Persistent URL 服务等开源项目的创始成员。

当前，他是马里兰大学研究所高级计算机研究协会MIND实验室的驻校企业家。他正在带领团队实现Policy-Aware Web项目，该项目为万维网开发了下一代访问控制系统。戴维创建了Tucana技术公司，这是一个语义网数据库供应商，2005年被诺斯罗普–格鲁曼（Northrop Grumman）公司收购。在创办Tucana之前，戴维于1995~2002年在澳大利亚创建了Plugged In软件公司，这也是一家成功的软件服务公司。

戴维还在玛丽华盛顿大学任计算机科学的兼职讲师，并在澳大利亚昆士兰大学研究重组数据技术对软件维护的应用。

**Joe Zenevitch（New York，New York，U.S.）**
**乔·泽尼维奇（美国纽约州纽约市）**

乔·泽尼维奇是ThoughtWorks公司的高级项目经理。在该公司他为最先进的软件开发项目提供程序和项目管理服务，同时还做商业分析和敏捷技术培训工作。乔有超过20年的软件开发经验，做过15个项目的管理工作。他有传统项目管理方法的知识背景，自从ThoughtWorks公司在1998年在项目中使用敏捷技术以来，乔就专门研究敏捷项目管理。

乔的联系方式是joez@thoughtworks.com。

［Ｇｅｎｅｒａｌ Ｉｎｆｏｒｍａｔｉｏｎ］
书名＝项目经理应该知道的９７件事
作者＝（美）戴维斯编
页数＝２１６
出版社＝人民邮电出版社
出版日期＝２０１１．０７
ＳＳ号＝１２７９５８７９
ＤＸ号＝０００００８１３１３９５
ＵＲＬ＝ｈｔｔｐ：／／ｂｏｏｋ２．ｄｕｘｉｕ．ｃｏｍ／ｂｏｏｋＤｅｔａｉｌ．ｊｓｐ？ｄｘＮｕｍｂｅｒ＝０００００８１
３１３９５＆ｄ＝８５８６６Ｂ０Ｅ７５２６７９６Ａ７２１ＦＥ６２１５９０Ａ３Ａ１Ｂ